Università degli Studi di Milano-Bicocca
Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea Magistrale in Informatica

Tesi di Laurea

Continuos Time Bayesian Networks Classifiers

Sottotitolo

Candidato:
Leonardo Di Donato
Matricola 744739

Relatore:
Prof. F. Antonio Stella

Correlatore:
Dott. Daniele Codecasa

Anno Accademico 2012-2013

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

— Oscar Wilde

Dedicato a tutti gli appassionati di LaTeX.

INDICE

ELENCO DELLE FIGURE

ELENCO DELLE TABELLE

ELENCO DEGLI ALGORITMI

SOMMARIO

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

ABSTRACT

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis.
Curabitur dictum gravida mauris.

— Donald Ervin Knuth

RINGRAZIAMENTI

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Milano, settembre 2013 L.

INTRODUZIONE

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

IL PRIMO CAPITOLO offre una visione d'insieme della storia di LaTeX e ne vengono presentate le idee di fondo.

IL SECONDO CAPITOLO offre una visione d'insieme della storia di LaTeX e ne vengono presentate le idee di fondo.

IL TERZO CAPITOLO spiega le operazioni, veramente semplici, per installare LaTeX sul proprio calcolatore.

IL QUARTO CAPITOLO descrive sinteticamente le principali norme tipografiche della lingua italiana, utili nella composizione di articoli, tesi o libri.

IL QUINTO CAPITOLO descrive sinteticamente le principali norme tipografiche della lingua italiana, utili nella composizione di articoli, tesi o libri.

IL SESTO CAPITOLO descrive sinteticamente le principali norme tipografiche della lingua italiana, utili nella composizione di articoli, tesi o libri.

IL SETTIMO CAPITOLO descrive sinteticamente le principali norme tipografiche della lingua italiana, utili nella composizione di articoli, tesi o libri.

L'APPENDICE A descrive sinteticamente le principali norme tipografiche della lingua italiana, utili nella composizione di articoli, tesi o libri.

1 CONTINUOS TIME BAYESIAN NETWORK

In questo capitolo si introducono i concetti fondamentali relativi alle Continuos time Bayesian Network (CTBN). Le CTBN sono un framework capace di modellare processi stocastici a tempo continuo e con spazio degli stati discreto.

Prima di affrontare tale argomento si presentano alcuni concetti propedeutici a questo lavoro di tesi: le Bayesian Network (BN) e i processi di Markov (sezione 1.1).

1.1 FONDAMENTI

Le Continuos time Bayesian Network utilizzano concetti e idee provenienti da teorie afferenti l'area statistica e del machine learning. Al fine di conferire alla discussione sulle CTBN un quadro iniziale completo ed esauriente, si presentano quindi gli aspetti di maggior rilievo di tali argomenti.

BAYESIAN NETWORK

Le Continuos time Bayesian Network utilizzano una rappresentazione strutturata dello spazio degli stati propria della teoria delle Bayesian Network. Ne ereditano perciò gli aspetti chiave (e. g., indipendenza condizionale) nonché l'insieme delle tecniche algoritmiche per l'apprendimento e l'inferenza.

PROCESSI DI MARKOV

Le Continuos time Bayesian Network descrivono la dinamica evolutiva di variabili casuali tramite un costituito da un insieme di processi di Markov condizionali.

1.1.1 Bayesian Network

Una Bayesian Network è un modello grafico probabilistico costituito da un grafo aciclico orientato (DAG)[1]. I nodi di tale grafo rappresentano un insieme di variabili casuali mentre gli archi evidenziano le dipendenze (e le indipendenze) condizionali fra esse (Korb e Nicholson, 2011). Una BN rappresenta la distribuzione di probabilità congiunta del suo insieme di variabili casuali tramite la distribuzione di probabilità condizionale di ognuna di essa (si veda l'equazione 1.2). Le

1 Un grafo aciclico orientato (anche detto grafo aciclico diretto o digrafo aciclico) è un tipo di grafo che non ammette cicli ed i cui archi sono orientati: comunque si scelga un vertice non è possibile tornare ad esso percorrendo gli archi del grafo.

BN sono quindi modelli grafico probabilistici con cui è possibile modellare in modo probabilistico le relazioni causali tra variabili. Esse risultano molto utili nella rappresentazione e analisi di domini caratterizzati da incertezza. Sono infatti usate in svariate applicazioni di supporto alle decisioni, bioinformatica, biologia computazionale, data mining, information retrieval e classificazione.

Rappresentazione

Di seguito si fornisce la definizione formale delle Bayesian Network e si introducono i loro aspetti basilari.

Definizione 1.1 (Bayesian Network)**.** Una Bayesian Network $\mathcal{B}$ è una coppia $\mathcal{B} = (\mathcal{G}, \boldsymbol{\theta}_{\mathcal{G}})$ costituita da:

- $\mathcal{G} = (\mathbf{V}(\mathcal{G}), \mathbf{A}(\mathcal{G}))$, un grafo aciclico orientato dove:
  - $\mathbf{V}(\mathcal{G}) = \{V_1, \ldots, V_n\}$ è l'insieme dei nodi, ognuno dei quali è associato ad una distribuzione di probabilità condizionale (CPD)[2]
  - $\mathbf{A}(\mathcal{G}) \subseteq \mathbf{V}(\mathcal{G}) \times \mathbf{V}(\mathcal{G})$ è l'insieme degli archi fra i nodi $\mathbf{V}(\mathcal{G})$
- $\boldsymbol{\theta}_{\mathcal{G}}$, insieme delle CPD dei nodi che specifica $\mathbf{P}_{\mathcal{B}}$, la distribuzione di probabilità congiunta delle variabili casuali $\mathbf{X}_{\mathbf{V}(\mathcal{G})}$ a cui corrispondono i nodi $\mathbf{V}(\mathcal{G})$.

Osservazione 1.1.1. Ogni nodo di una BN è condizionalmente indipendente (si veda definizione 1.2) dai suoi non–discendenti dati i suoi nodi genitori.

La CPD di ogni variabile casuale $X_i \in \mathbf{X}_{\mathbf{V}(\mathcal{G})}$ esprime i suoi valori di probabilità in funzione dei valori assunti da $\mathrm{Pa}(X_i)$, notazione con cui si denota l'*insieme dei nodi genitori* per ogni nodo o variabile casuale.

Un arco da un nodo genitore verso un nodo figlio di $\mathcal{G}$ rappresenta una dipendenza diretta fra le corrispettive variabili casuali (si veda Russell e Norvig, 2003, sezione 14.1). I nodi non direttamente connessi rappresentano variabili casuali condizionalmente indipendenti dagli altri nodi (per quanto riguarda il concetto di *indipendenza condizionale* si rimanda alla definizione 1.2).

Prima di procedere con la discussione si introduce la Chain Rule, proprietà fondamentale delle BN.

Teorema 1.1 (Chain Rule)**.** *Dato un insieme di variabili casuali e una distribuzione di probabilità congiunta definita su di esse è possibile calcolare qualsiasi elemento di tale distribuzione tramite le distribuzioni di probabilità condizionale delle variabili casuali.*

2 Nel caso di variabili causali discrete, le CPD sono rappresentabili come delle tabelle che contengono i valori di probabilità di un nodo in funzione di tutte le possibili configurazioni dei nodi genitori (cioè l'insieme dei nodi da cui parte un arco che punta al nodo di interesse). Tali tabelle sono spesso chiamate tabelle di probabilità condizionale (CPT).

Perciò, dato un insieme di variabili casuali $A_1, \ldots, A_n$ è possibile calcolare il valore di tale membro della distribuzione di probabilità congiunta applicando la definizione di probabilità condizionale:

$$\mathbf{P}(A_1, \ldots, A_n) = \mathbf{P}(A_n \mid A_{n-1}, \ldots, A_1) \cdot \mathbf{P}(A_{n-1}, \ldots, A_1).$$

Ripetendo tale processo per ogni termine finale si ottiene:

$$\mathbf{P}\Big(\bigcap_{k=1}^{n} A_k\Big) = \prod_{k=1}^{n} \mathbf{P}\Big(A_k \mid \bigcap_{j=1}^{k-1} A_j\Big). \tag{1.1}$$

Applicando l'equazione 1.1 alle Bayesian Network si dice che la distribuzione di probabilità congiunta $\mathbf{P}_{\mathcal{B}}$ si *fattorizza* rispetto al grafo $\mathcal{G}$ se è possibile scrivere:

$$\mathbf{P}_{\mathcal{B}}(X_1, \ldots, X_n) = \prod_{i=1}^{n} \mathbf{P}(X_i \mid \mathrm{Pa}(X_i)). \tag{1.2}$$

L'equazione 1.2 esprime quindi la *proprietà di fattorizzazione* della distribuzione congiunta del modello grafico, detta *distribuzione di probabilità globale*, ed è ciò che permette di descriverla efficientemente in funzione delle distribuzioni condizionali dei nodi (Russell e Norvig, 2003, sezione 14.2), dette *distribuzioni di probabilità locali*. Questa proprietà contiene in sè il concetto di *proprietà di Markov* (si veda la definizione 1.3), il quale attesta che ogni nodo di una Bayesian Network dipende solo ed esclusivamente dai suoi nodi genitori (Korb e Nicholson, 2011, sottosezione 2.2.4). Si noti inoltre, che le Bayesian Network richiedono (DAG, definizione 1.1) che la loro componente $\mathcal{G}$ non contenga cicli affinché possano rispettare tale proprietà (Russell e Norvig, 2003, sezione 14.1).

Poiché, come detto, una Bayesian Network stabilisce che ogni nodo, dati i suoi genitori, è *condizionalmente indipendente* da ogni altro nodo che non sia un suo discendente, di seguito si introduce tale concetto formalmente.

Definizione 1.2 (Indipendenza condizionale)**.** Un evento A è *condizionalmente indipendente* da un evento B, data l'evidenza su un evento C, qualora la conoscenza di B non apporta alcuna variazione alla probabilità di A rispetto a quella conseguente alla conoscenza di C. Formalmente, ciò significa che:

$$\mathbf{P}(A, B \mid C) = \mathbf{P}(A \mid B, C) \cdot \mathbf{P}(B \mid C) = \mathbf{P}(A \mid C) \cdot \mathbf{P}(B \mid C).$$

Da cui segue che:

$$A \perp B \mid C \iff \mathbf{P}(A \mid B, C) = \mathbf{P}(A \mid C).$$

In termini non formali, supponendo di essere nel caso della definizione, cioè di avere una variabile casuale A *condizionalmente indipendente* da B dato C, ciò significa che è possibile ignorare B poiché essa

non ha alcun riflesso sulla distribuzione condizionale di A quando sia noto l'evento C.

Si noti che il concetto appena espresso gioca un ruolo importante per i modelli probabilistici, quali sono le Bayesian Network, semplificando i calcoli richiesti per l'inferenza e l'apprendimento. Le Bayesian Network ereditano questi benefici dell'indipendenza condizionale come conseguenza della loro definizione (si veda l'osservazione 1.1.1). Infatti, la distribuzione condizionale di ogni variabile casuale X_i dipende solo ed esclusivamente dal valore dei suoi genitori, $\mathrm{Pa}(X_i)$, mentre ignora completamente i valori dei nodi che non discendono da essa, $\mathrm{Nd}(X_i)$.

Grazie alla definizione 1.2 è possibile esprimere in modo formale il concetto appena espresso per ogni nodo $X_i \in \mathbf{X}_{\mathbf{V}(\mathcal{G})}$:

$$\mathbf{P}(X_i \mid E, \mathrm{Pa}(X_i)) = \mathbf{P}(X_i \mid \mathrm{Pa}(X_i)) \quad \forall\, E \in \mathrm{Nd}(X_i),$$

dove $\mathrm{Nd}(X_i)$ è l'insieme dei nodi non–discendenti (ed E è una variabile casuale o un insieme di variabili casuali ad essi associati). In base a ciò si dice quindi che le Bayesian Network rispettano l'*assunzione locale di Markov*.

Apprendimento e Inferenza

In questa sezione si descrivono brevemente e a scopo introduttivo i processi di apprendimento e inferenza sulle Bayesian Network.

Il problema dell'apprendimento per le Bayesian Network si divide principalmente in due casi:

- apprendere le CPD, nota la struttura
- apprendere sia le CPD, sia la struttura (incognita).

In entrambi i casi è di grande aiuto la rappresentazione efficiente delle Bayesian Network che, tramite la *fattorizzazione* della distribuzione di probabilità congiunta, permette di rappresentarla in modo compatto (tramite l'equazione 1.2) riducendo notevolmente il numero di parametri da calcolare.

Come detto, per specificare completamente una Bayesian Network è necessario rappresentare completamente la distribuzione di probabilità congiunta delle sue variabili tramite la distribuzione di probabilità condizionale di ognuna di esse. In generale, tali distribuzioni condizionali possono avere una qualsiasi forma anche se, al fine di semplificare i calcoli, è comune utilizzare distribuzioni discrete o Gaussiane per modellarle. Nel caso in cui i dati siano parzialmente osservabili solitamente si procede tramite l'algoritmo di Expectation Maximization (EM), il quale alterna il calcolo dei valori attesi delle variabili casuali non osservate condizionalmente ai dati osservati con la massimizzazione della likelihood. Tale approccio generalmente converge ai valori di massima probabilità a posteriori per i parametri (si veda Dempster *et al.*, 1977).

Per l'*apprendimento dei parametri* esistono comunque una varietà di altri approcci possibili (si veda Heckerman, 1996) (e. g., trattare i parametri come variabili casuali sconosciute addizionali) che tuttavia non sono argomento di questo lavoro di tesi.

Si noti che le Bayesian Network non sono solamente un *modello discriminativo ma anche generativo* poiché possono essere utilizzate per soddisfare query arbitrarie, cioè per effettuare *inferenza probabilistica*: calcolare la distribuzione a posteriori di un insieme di variabili casuali data l'osservazione (evidenza) di altre (sfruttando il *teorema di Bayes*). In letteratura (si veda Heckerman, 1996) sono stati esplorati molti metodi di *inferenza esatta*, quali ad esempio l'eliminazione tramite integrazione o somma delle variabili non osservate che non fanno parte della query probabilistica o il metodo *clique tree propagation*. Questi metodi, come gli altri presenti in letteratura, sono sempre esponenziali rispetto al *tree-width*[3] del grafo. Per quanto riguarda invece gli algoritmi di *inferenza approssimata* si citano due tra i più comuni: l'*importance sampling* (Shachter e Peot, 1990) e i metodi Markov Chain Monte Carlo (MCMC) (*Gibbs sampling*, *Metropolis sampling*, e *Hybrid Monte Carlo sampling*), basati sul campionamento stocastico (si veda S. Geman e D. Geman, 1984; Gilks *et al.*, 1996; MacKay, 1998).

Nel caso in cui non si disponga della struttura di una BN è richiesto l'*apprendimento strutturale*. Gli algoritmi per l'apprendimento strutturale delle Bayesian Network possono essere divisi in due famiglie.

ALGORITMI BASATI SU VINCOLI

Algoritmi che apprendono la struttura del grafo analizzando le relazioni probabilistiche derivanti dalla proprietà di Markov tramite test di indipendenza condizionale e costruendo un grafo che soddisfi le proprietà di *d-separazione*[4] corrispondenti. I modelli risultanti sono spesso interpretati come *modelli causali* (Pearl, 1988).

ALGORITMI BASATI SU PUNTEGGIO

Algoritmi che assegnano un punteggio (tramite una funzione di scoring) a tutte le strutture candidate e utilizzando tecniche di ottimizzazione cercano di raggiungere il punteggio massimo. Gli algoritmi di ricerca *greedy* sono la scelta più comune, tuttavia qualsiasi procedura di ricerca può essere usata.

Gli *algoritmi basati su vincoli* sono basati sull'algoritmo Inductive Causation (IC) di Verma e Pearl (1991) che fornisce un contesto teo-

3 In teoria dei grafi, il *tree-width* è un numero associato ad un grafo. Esso corrisponde alla lunghezza minima di tutti i possibili alberi di decomposizione del grafo in esame. La lunghezza di un albero di decomposizione corrisponde alla dimensione massima dei suoi nodi, cioè sottoinsiemi dell'insieme dei vertici del grafo, sottratto 1.

4 Concetto di separazione direzionale tra insiemi di nodi collegato al concetto di *indipendenza condizionale*. Ad esempio, quando un insieme di nodi $\mathbf{E}$ *d-separa* un insieme di nodi $\mathbf{X} = \{A, B\}$ allora A e B sono condizionalmente indipendenti dato $\mathbf{E}$.

rico finalizzato all'apprendimento delle strutture dei modelli causali. L'algoritmo IC può essere riassunto nei tre passi successivi.

- Apprendimento dello scheletro (i. e., grafo non diretto) della rete. Poiché la ricerca esaustiva non è, nella maggior parte dei casi, computazionalmente realizzabile, tutti gli algoritmi di apprendimento restringono la ricerca al *Markov blanket*[5] di ogni nodo.
- Impostare la direzione degli archi che fanno parte di una *v-structure*[6].
- Impostare la direzione degli archi fra i nodi rimanenti affinché il vincolo di aciclicità sia rispettato.

Gli *algoritmi basati su punteggio* sono invece delle applicazioni dei vari algoritmi di ricerca euristica (e. g., *hill climbing*, *tabu search*, *best first search*, *simulated annealing*) che utilizzano una funzione di *scoring*. Solitamente la funzione di *scoring* è basata sulla *likelihood*, ovvero sulla la probabilità a posteriori dell'insieme dei dati di apprendimento (i. e., *training set*), data la struttura in esame e i parametri del modello. Tale funzione è spesso *score-equivalent*, affinché reti che definiscono la stessa distribuzione di probabilità abbiano lo stesso score (Chickering, 2013). Tuttavia, per quanto questi algoritmi siano utilizzati molto frequentemente, essi sono esponenziali rispetto al numero di nodi della struttura del grafo. Inoltre, qualora si utilizzi una strategia di ricerca locale, è probabile che l'algoritmo restituisca come risultato un minimo locale. Si fa notare che è possibile ridurre il tempo necessario richiesto per l'apprendimento strutturale fissando un numero massimo di genitori candidati e cercando esaustivamente in insiemi di tale cardinalità una struttura che massimizzi l'informazione mutua fra variabili (Heckerman *et al.*, 1995).

1.1.2 Processi di Markov

Sempre al fine di preparare la discussione delle Continuos time Bayesian Network si prosegue presentando alcuni concetti propedeutici relativi ai processi di Markov, una categoria di processi stocastici con assenza di memoria (Loève, 1978).

Definizione 1.3 (Proprietà di Markov)**.** Secondo la proprietà di Markov gli stati futuri di un processo stocastico sono indipendenti dagli stati passati, avendo evidenza sullo stato presente di tale processo.

5 Il *Markov blanket* di un nodo A è un insieme composto dai nodi genitori di A, dai suoi nodi figli e da tutti i nodi che condividono un figlio con A.

6 Una *v-structure* è una tripla di nodi $X_j \to X_i \leftarrow X_k$ incidenti su una connessione convergente.

Formalmente, un processo stocastico X gode di tale proprietà, se e solo se vale la seguente equazione (Loève, 1978):

$$\mathbf{P}(X(t+\Delta t)\,|\,X(t),\,X(s)) = \mathbf{P}(X(t+\Delta t)\,|\,X(t)), \tag{1.3}$$

per ogni s, e t tali che $s < t < \infty$.

I modelli che rispettano tale proprietà sono detti modelli che rispettano l'*assunzione di Markov*.

Di conseguenza la distribuzione di probabilità condizionale degli stati futuri di un processo stocastico che gode di tale proprietà è indipendente dagli stati passati dato quello attuale.

In altri termini ciò indica che lo stato futuro di una variabile casuale è *condizionalmente indipendente* (si veda la definizione 1.2) dalla sequenza dei suoi stati passati, avendo evidenza sul suo stato presente.

Dalla proprietà di Markov deriva la definizione dei processi di Markov.

Definizione 1.4 (Processo di Markov)**.** Si definisce (Loève, 1978) come processo di Markov un processo stocastico che gode della proprietà di Markov.

Definizione 1.5 (Catena di Markov)**.** Un processo di Markov che può assumere solo un numero finito di stati è solitamente definito come una *catena di Markov* (si veda Norris, 1998, p. 10).

Esistono due tipi di processi di Markov: omogenei e non. Si procede quindi fornendone le definizioni.

Definizione 1.6 (Processo di Markov omogeneo)**.** Un processo di Markov è detto *omogeneo* qualora $\mathbf{P}(X(t+\Delta t)\,|\,X(t))$ non dipenda dal tempo t. Affinché ciò sia vero deve risultare che:

$$\mathbf{P}(X(t+\Delta t)\,|\,X(t)) = \mathbf{P}(X(\Delta t)\,|\,X(0)). \tag{1.4}$$

Data quindi una variabile casuale X e l'insieme delle sue istanziazioni $\mathrm{val}(X) = \{x_1, \ldots, x_J\}$, $X(t)$ è un processo di Markov *omogeneo, a tempo continuo e stati finiti* se e solo se la sua dinamica è definibile in termini di:

- una distribuzione di probabilità iniziale $\mathbf{P}^0_X$ su $\mathrm{val}(X)$
- una matrice di intensità $\mathbf{Q}_X$.

Definizione 1.7 (Matrice di intensità)**.** Una matrice di intensità (IM), rappresenta un *modello di transizione Markoviano*:

$$\mathbf{Q}_X = \begin{bmatrix} -\mathbf{q}_{x_1} & \mathbf{q}_{x_1x_2} & \cdots & \mathbf{q}_{x_1x_K} \\ \mathbf{q}_{x_2x_1} & -\mathbf{q}_{x_2} & \cdots & \mathbf{q}_{x_2x_K} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{q}_{x_Kx_1} & \mathbf{q}_{x_Kx_2} & \cdots & -\mathbf{q}_{x_K} \end{bmatrix}.$$

Lo scopo di un matrice di intensità è descrivere il comportamento transiente di X, un processo di Markov omogeneo.

Osservazione 1.7.1. L'*ordine* di una matrice di intensità corrisponde a $K = |val(X)|$, la cardinalità dell'insieme dei valori assunti da X.

Affinché $\mathbf{Q}_X$ sia una matrice di intensità valida, ogni sua riga deve sommare a 0:

$$\mathbf{q}_{x_i} = \sum_{i \neq j} \mathbf{q}_{x_i x_j} \quad \text{con} \quad \mathbf{q}_{x_i}, \mathbf{q}_{x_i x_j} > 0.$$

Data quindi una matrice di intensità $\mathbf{Q}_X$ essa descrive il comportamento transiente di $X(t)$. Se $X(0) = x_i$, allora il processo di Markov omogeneo (e indicizzato dal tempo t) $X(t)$ rimarrà nello stato x_i una quantità di tempo *esponenzialmente distribuita* rispetto al parametro $\mathbf{q}_{x_i}$. Di conseguenza la *funzione di densità* f e la corrispondente *funzione di ripartizione*[7] F sono quelle della distribuzione esponenziale:

$$\begin{aligned} f(t) &= \mathbf{q}_{x_i} e^{-\mathbf{q}_{x_i} t}, \quad t > 0 \\ F(t) &= 1 - e^{-\mathbf{q}_{x_i} t}, \quad t \geqslant 0. \end{aligned} \tag{1.5}$$

Quando un modello di transizione è definito esclusivamente tramite una matrice di intensità $\mathbf{Q}_X$ si dice che esso usa una *parametrizzazione pura* delle intensità. In tal caso i parametri per un processo di Markov omogeneo con K stati sono $\{\mathbf{q}_{x_i}, \mathbf{q}_{x_i x_j} : 1 \leqslant i, j \leqslant K, i \neq j\}$.

Mentre gli elementi sulla diagonale di una matrice di intensità, $\mathbf{q}_{x_i}$, codificano una quantità che può essere interpretata come la *"probabilità istantanea"* che X abbandoni lo stato x_i, gli elementi non sulla diagonale, $\mathbf{q}_{x_i x_j}$, esprimono l'*intensità di transizione* dallo stato x_i allo stato x_j.

Tuttavia, questa non è l'unica parametrizzazione possibile per un processo di Markov omogeneo. Si noti infatti che la distribuzione di probabilità locale sulle transizioni di X è fattorizzata in due parti.

Definizione 1.8 (Parametrizzazione mista delle intensità)**.** La *parametrizzazione mista* delle intensità per un processo di Markov omogeneo X con K stati è composta da due insiemi di parametri:

$$\begin{aligned} \mathbf{q}_X &= \{\mathbf{q}_{x_i} : 1 \leqslant i \leqslant K\}, \\ \theta_X &= \{\theta_{x_i x_j} : 1 \leqslant i, j \leqslant K, i \neq j\}. \end{aligned}$$

La semantica di tali insiemi di parametri è la seguente:

- $\mathbf{q}_X$ è un insieme di intensità $\mathbf{q}_{x_i}$ che parametrizzano una *distribuzione di probabilità esponenziale* ed esprimono quando avvengono le transizioni

7 Nel calcolo delle probabilità la funzione di ripartizione di una variabile casuale X a valori reali, anche nota come funzione di distribuzione cumulativa, è la funzione che associa a ciascun valore x la probabilità che X assuma valori minori o uguali ad x (i. e., $\mathbf{P}(X \leqslant x)$).

- θ_X è un insieme di probabilità $\theta_{x_i x_j}$ che rappresentano la *probabilita di transitare* dallo stato x_i allo stato x_j, con $i \neq j$, sapendo che avverrà un salto ad un determinato istante di tempo.

Osservazione 1.8.1. Si osservi che, aldilà del tipo di parametrizzazione con cui si sceglie di definire un modello di transizione, il numero di parametri necessari è pari a K^2 sebbene il numero di parametri liberi sia solo $K^2 - K$ (Nodelman, 2007).

Osservazione 1.8.2. Si noti, inoltre, che una parametrizzazione può essere più chiara dell'altra a seconda del processo in cui si è coinvolti, di conseguenza nel prosieguo le si utilizzerà entrambe in modo intercambiabile.

Al fine di correlare questi due tipi di parametrizzazione dei modelli di transizione si riporta il seguente teorema (Nodelman, 2007).

Teorema 1.2. *Dati* X *e* Y*, due processi di Markov omogenei con lo stesso spazio degli stati e la stessa distribuzione di probabilità iniziale, se il modello di transizione di* X *è definito tramite la matrice di intensità* $\mathbf{Q}_X$ *e quello di* Y *è definito tramite la parametrizzazione mista* $\boldsymbol{q}_Y$, θ_Y*, allora* X *e* Y *sono stocasticamente equivalenti*[8] *solo se:*

$$\boldsymbol{q}_{y_i} = \boldsymbol{q}_{x_i}$$

e

$$\theta_{y_i y_j} = \frac{\boldsymbol{q}_{x_i x_j}}{\boldsymbol{q}_{x_i}}.$$

Osservazione 1.8.3. Si osservi che il teorema 1.2 formalizza la relazione che sussiste fra i parametri $\mathbf{q}$ e θ.

Quindi, qualsiasi sia la parametrizzazione utilizzata per rappresentare il modello di transizione di un processo di Markov omogeneo X, è possibile calcolare:

- il *tempo atteso* di una transizione uscente dallo stato x_i

$$1/\mathbf{q}_{x_i}$$

- la *"probabilità istantanea"* di transizione dallo stato x_i allo stato x_j sapendo che avverrà un salto ad un determinato istante di tempo

$$\theta_{x_i x_j} = \mathbf{q}_{x_i x_j}/\mathbf{q}_{x_i}.$$

Infine, si noti che la matrice $\mathbf{Q}_X$ fa in modo che X soddisfi la proprietà di Markov poiché il comportamento futuro di X è definito solamente in base al suo stato attuale (vale l'equazione 1.4).

8 Due processi di Markov sono detti *stocasticamente equivalenti* se posseggono lo stesso spazio degli stati e le stesse probabilità di transizione (Gihman e Skorohod, 1973).

Definizione 1.9 (Processo di Markov condizionale)**.** Un processo di Markov le cui intensità di transizione variano nel tempo non in funzione del tempo ma in funzione dei valori assunti ad ogni determinato istante t da un insieme di altre variabili, che evolvono anch'esse come dei processi di Markov, è detto essere un processo di Markov condizionale (o processo di Markov non omogeneo).

Assumendo quindi che una variabile casuale X evolva come un processo di Markov $X(t)$ e che la sua dinamica sia condizionata da un insieme di altre variabili casuali $Pa(X)$, anch'esse dei processi di Markov, è possibile definire per tale variabile casuale una matrice di intensità condizionale (CIM) $\mathbf{Q}_{X \mid Pa(X)}$.

Specificando una distribuzione di probabilità iniziale su X si definisce quindi un processo di Markov il cui comportamento dipende dalle istanziazioni dei valori di $Pa(X)$.

Definizione 1.10 (Matrice di intensità condizionale)**.** Dato un insieme di processi di Markov $Pa(X)$, una matrice di intensità condizionale $\mathbf{Q}_{X \mid Pa(X)}$ è costituita da un insieme di matrici di intensità $\mathbf{Q}_{X \mid pa_i(x)}$, una per ogni diversa istanziazione $pa_i(x)$ di $Pa(X)$ (Stella e Amer, 2012):

$$\mathbf{Q}_{X \mid Pa(X)} = \left\{ \mathbf{Q}_{X \mid pa_1(x)}, \mathbf{Q}_{X \mid pa_2(x)}, \dots, \mathbf{Q}_{X \mid pa_n(x)} \right\}.$$

Ogni matrice di intensità di $\mathbf{Q}_{X \mid Pa(X)}$ è del seguente tipo:

$$\mathbf{Q}_{X \mid pa_i(x)} = \begin{bmatrix} -\mathbf{q}_{x_1}^{pa_i(x)} & \mathbf{q}_{x_1 x_2}^{pa_i(x)} & \cdots & \mathbf{q}_{x_1 x_K}^{pa_i(x)} \\ \mathbf{q}_{x_2 x_1}^{pa_i(x)} & -\mathbf{q}_{x_2}^{pa_i(x)} & \cdots & \mathbf{q}_{x_2 x_K}^{pa_i(x)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{q}_{x_K x_1}^{pa_i(x)} & \mathbf{q}_{x_K x_2}^{pa_i(x)} & \cdots & -\mathbf{q}_{x_K}^{pa_i(x)} \end{bmatrix}.$$

Di seguito si presenta un breve esempio finalizzato alla comprensione pratica delle matrici di intensità condizionali (CIM) e del loro scopo.

Esempio 1.10.1.
Date due variabili causali, $E(t)$ e $H(t)$, delle quali la prima modella l'eventualità che un individuo stia mangiando o meno (se $e = 2$ allora l'individuo sta mangiando, viceversa se $e = 1$) mentre la seconda modella l'eventualità che lo stesso individuo abbia fame o meno (se $h = 2$ allora l'individuo è affamato, viceversa se $h = 1$) e la matrice di intensità condizionale $\mathbf{Q}_{E \mid H}$, che è un insieme composto dalle matrici di intensità $\mathbf{Q}_{E \mid h=1}$ e $\mathbf{Q}_{E \mid h=2}$, è possibile calcolare la probabilità degli eventi della variabile casuale E condizionatamente all'evidenza che si possiede sulla variabile casuale H.

$$\mathbf{Q}_{E \mid h=1} = \begin{matrix} \\ 1 \\ 2 \end{matrix} \begin{matrix} \begin{matrix} 1 & 2 \end{matrix} \\ \begin{bmatrix} -.01 & .01 \\ 10 & -10 \end{bmatrix} \end{matrix} \qquad \mathbf{Q}_{E \mid h=2} = \begin{matrix} \\ 1 \\ 2 \end{matrix} \begin{matrix} \begin{matrix} 1 & 2 \end{matrix} \\ \begin{bmatrix} -2 & 2 \\ .01 & -.01 \end{bmatrix} \end{matrix}.$$

Ipotizzando che l'unità temporale corrisponda a un'ora:

- un individuo affamato ($h = 2$) che non sta mangiando ($e = 1$) inizierà a mangiare in 30 minuti poiché

$$\frac{1}{\mathbf{q}_{e=1\,|\,h=2}} = \frac{1}{2},$$

- un individuo non affamato ($h = 1$) che sta mangiando ($e = 2$) smetterà di mangiare ($e = 1$) entro 6 minuti poiché

$$\frac{1}{\mathbf{q}_{e=2\,|\,h=1}} = \frac{1}{10};$$

si osservi che la *"probabilità istantanea"* di transizione da $e = 2$ a $e = 1$ è

$$\theta_{e=2,e=1\,|\,h=1} = \frac{\mathbf{q}_{e=2,e=1\,|\,h=1}}{\mathbf{q}_{e=2\,|\,h=1}} = \frac{10}{10} = 1,$$

ciò poiché le matrici di intensità hanno dimensione 2×2 e, dovendo ogni loro riga sommare a 0, gli elementi sulla diagonale sono uguali al rispettivo (stessa riga) e unico elemento non sulla diagonale.

1.2 DEFINIZIONI PRELIMINARI

Nelle precedenti sezioni sono stati illustrati i concetti che si pongono a fondamento delle Continuos time Bayesian Network:

- le Bayesian Network: utili a comprendere la rappresentazione strutturata dello spazio degli stati delle CTBN, l'utilizzo della nozione di indipendenza condizionale e le conseguenti tecniche di apprendimento e inferenza
- i processi di Markov, omogenei e non, al fine di introdurre le modalità di rappresentazione (qualitativa e quantitativa) delle CTBN.

Prima di presentare le Continuos time Bayesian Network come una collezione di processi di Markov a tempo continuo non omogenei e con spazio degli stati discreto (Nodelman, 2007), si forniscono alcune definizioni utili per il prosieguo della discussione.

Definizione 1.11 (Variabile di processo)**.** Una variabile di processo $\mathbf{X}$, anche detta Process Variable (PV) (Nodelman, 2007), è un insieme di processi di Markov a tempo continuo $X(t)$.

Definizione 1.12 (Traiettoria)**.** Istanziazione di un insieme di valori per $X(t)$ al variare di t.

Definizione 1.13 (J-time-segment)**.** Partizionamento di un intervallo temporale $[0, T)$ in J intervalli chiusi a sinistra:

$$[0, t_1); [t_1, t_2); \dots; [t_{J-1}, T).$$

Nota 1.13.1. È possibile riferirsi a tale concetto anche tramite l'espressione *"insieme dei segmenti temporali"*.

Definizione 1.14 (J-evidence-stream)**.** Data una variabile di processo $\mathbf{X}$ composta da N variabili casuali e un insieme di segmenti temporali composto da J intervalli, un J-*evidence-stream* è l'insieme delle istanziazioni comuni $\mathbf{X} = \mathbf{x}$ associate ad ogni intervallo temporale per ogni sottoinsieme delle variabili casuali (Stella e Amer, 2012). È denotato con $(\mathbf{X}^1 = \mathbf{x}^1, \mathbf{X}^2 = \mathbf{x}^2, \dots, \mathbf{X}^J = \mathbf{x}^J)$, o più concisamente con $(\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^J)$.

Nota 1.14.1. È possibile riferirsi a tale concetto anche tramite l'espressione *"flusso di evidenze"*.

Nota 1.14.2. Un flusso di evidenze $(\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^J)$. è detto essere *completamente osservato* se lo stato di tutte le variabili $X_n \in \mathbf{X}$ è conosciuto in tutto l'intervallo $[0, T)$. Viceversa, un flusso di evidenze è detto *parzialmente osservato*.

1.3 RAPPRESENTAZIONE

Una Continuos time Bayesian Network è un modello grafico in cui ogni nodo rappresenta una variabile casuale i cui stati evolvono in modo continuo bel tempo. Le dinamiche evolutive degli stati dei nodi sono governate e dipendono dal valore che gli stati dei nodi padre[9] assumono (Nodelman, 2007). Quindi ogni nodo è un processo di Markov condizionale (si veda la definizione 1.9) a tempo continuo e spazio degli stati discreto.

Una CTBN è composta principalmente da due componenti:

- una distribuzione di probabilità iniziale
- le componenti che regolano l'evoluzione nel tempo del sistema

Più formalmente si definisce:

Definizione 1.15 (Continuos time Bayesian Network)**.** Data una variabile di processo $\mathbf{X}$, insieme di processi di Markov $X_1, X_2, \dots, X_N$ a tempo continuo e con spazio degli stati finito $val(X_n) = \{x_1, \dots, x_J\}$ (dove $n = 1, \dots, N$), una CTBN $\mathcal{N}$ su $\mathbf{X}$ consiste di:

- una distribuzione di probabilità iniziale $\mathbf{P}^0_{\mathbf{X}}$ specificata come una Bayesian Network $\mathcal{B}$ su $\mathbf{X}$

9 Con il termine "nodo padre", o *parent node*, si intende un nodo il cui stato condiziona quello di un altro nodo del modello grafico.

- un modello di transizione a tempo continuo, specificato da:
  - un grafo $\mathcal{G}$, orientato e non necessariamente aciclico, composto dai nodi $X_1, X_2, \ldots, X_N$, ognuno dei quali possiede un insieme di genitori denotato da $\mathrm{Pa}(X_n)$
  - una matrice di intensità condizionale $\mathbf{Q}_{X_n \mid \mathrm{Pa}(X_n)}$ per ogni nodo $X_n \in \mathbf{X}$.

Per ogni variabile causale $X_n \in \mathbf{X}$ di $\mathcal{N}$ si ha quindi un insieme di modelli di probabilità locali: $\mathbf{Q}_{X_n \mid \mathrm{Pa}(X_n)}$, la CIM di X_n, è infatti un insieme di modelli di transizione Markoviani la cui cardinalità è pari a quella dell'insieme delle diverse istanziazioni di $\mathrm{Pa}(X_n)$.

Si riscontra, quindi, quanto già affermato in precedenza (si veda la sezione 1.1 a pagina 1), cioè che una CTBN esprime la sua dinamica evolutiva globale tramite un unico processo di Markov omogeneo, costituito da un insieme di processi di Markov condizionali (un insieme di CIM e relative distribuzioni di probabilità iniziali).

Si noti che, diversamente dalle Bayesian Network, nelle Continuos time Bayesian Network gli archi fra i nodi rappresentano le dipendenze nel tempo. Per tale motivo è possibile che la componente $\mathcal{G}$ del modello di transizione continuo contenga dei cicli. Tra l'altro, come vedremo nel prosieguo, la mancanza di tale vincolo di aciclicità porta a notevoli vantaggi computazionali relativamente all'apprendimento della struttura di una CTBN dai dati.

1.4 APPRENDIMENTO

In questa sezione si argomenta sulla probabilità di un *insieme di dati completo* rispetto a una Continuos time Bayesian Network. A tal fine si mostra come una CTBN possa essere decomposta in un aggregato di modelli di probabilità locali relativi alle singole variabili casuali e espressa in termini di *statistiche sufficienti* aggregate.

Si affronta infine il processo di apprendimento dei parametri delle Continuos time Bayesian Network da *dati completi*. I processi di apprendimento relativi a dati non completi sono tralasciati poiché non facenti parte degli argomenti di questo lavoro di tesi.

Definizione 1.16 (Insieme di dati completo)**.** Dato un insieme di variabili casuali, un insieme di dati $\mathcal{D} = \{\delta_1, \ldots, \delta_h\}$ si dice *completo* se ogni δ_i (con $i = 1, \ldots, h$) è un insieme di traiettorie completamente osservate delle variabili casuali (i. e., l'istanziazione di tutte le variabili casuali è osservabile per ogni istante temporale di ogni traiettoria).

1.4.1 Statistiche sufficienti

Le *statistiche sufficienti* per un singolo processo di Markov omogeneo $X(t)$ riassumono la sua dinamica evolutiva con:

- $T[x]$: la quantità di tempo trascorsa nello stato x
- $M[x, x']$: il numero di transizioni dallo stato x allo stato x'.

Il numero totale di transizioni uscenti da uno stato x è:

$$M[x] = \sum_{x'} M[x, x'].$$

Nel caso di un processo di Markov condizionale è invece necessario considerare anche l'istanziazione dell'insieme $Pa(X)$ dei nodi genitori:

- $T[x|pa_i(x)]$: la quantità di tempo trascorsa nello stato x quando $Pa(X) = pa_i(x)$
- $M[x, x'|pa_i(x)]$: il numero di transizioni dallo stato x allo stato x' quando $Pa(X) = pa_i(x)$.

Chiaramente, il numero totale di transizioni si calcola come sopra.

1.4.2 Likelihood

Al fine di presentare il calcolo della likelihood[10] di una CTBN rispetto a un dataset completo $\mathcal{D} = \{\delta_1, \ldots, \delta_h\}$ è bene procedere per gradi e iniziare presentando dapprima la likelihood di una singola transizione di un singolo processo di Markov omogeneo $X(t)$.

Likelihood di una singola transizione

Data una tripla $d = \langle x_d, t_d, x_{d'} \rangle \in \delta$, la quale esprime una transizione di $X(t)$ da x_d a $x_{d'}$ dopo che esso ha trascorso t_d tempo in x_d, è possibile scrivere la likelihood di questa singola transizione d in funzione dei parametri:

$$\begin{aligned} L_X(\mathbf{q}, \theta : d) &= L_X(\mathbf{q} : d) \cdot L_X(\theta : d) \\ &= \mathbf{q}_{x_d} e^{-\mathbf{q}_{x_d} t_d} \cdot \theta_{x_d x_{d'}}. \end{aligned} \tag{1.6}$$

Si noti che l'equazione 1.6 è ricavata moltiplicando la *funzione di distribuzione di probabilità* di $X(t)$ (equazione 1.5) per la *"probabilità istantanea"* di transizione (si veda la definizione 1.7).

10 La likelihood di un insieme di valori dei parametri, dato un insieme di dati, corrisponde alla probabilità dell'insieme dei dati, dati tali valori dei parametri.

Likelihood di un dataset completo

Poiché tutte le transizioni sono osservabili, la *likelihood* del dataset $\mathcal{D}$ può essere decomposta come un prodotto delle likelihood individuali di ogni singola transizione d (si veda Nodelman *et al.*, 2002, p. 3). Per tale motivo $\mathcal{D}$ è sintentizzabile aggregando le *statistiche sufficienti* relative a ogni processo di Markov condizionale di una CTBN.

Quindi la likelihood di un dataset completo $\mathcal{D}$ rispetto a un singolo processo di Markov omogeneo $X(t)$ è:

$$\begin{aligned} L_X(\mathbf{q}, \theta : \mathcal{D}) &= \Big[\prod_{d\in\mathcal{D}} L_X(\mathbf{q} : d)\Big]\Big[\prod_{d\in\mathcal{D}} L_X(\theta : d)\Big] \\ &= \Big[\prod_x (\mathbf{q}_x)^{M[x]} e^{-\mathbf{q}_x T[x]}\Big]\Big[\prod_x \prod_{x\neq x'} (\theta_{xx'})^{M[x,x']}\Big]. \end{aligned} \tag{1.7}$$

Si supponga ora di traslare questo concetto a una Continuos time Bayesian Network $\mathcal{N}$ con N nodi: per ogni nodo X_i, con $i = 1, \dots N$ è necessario considerare tutte le transizioni contestualmente all'istanziazione dell'insieme $Pa(X_i)$ dei suoi nodi genitori. Poiché, nel caso di *dati completi*, si conosce sempre l'istanziazione di $Pa(X_i)$, allora, per ogni istante di tempo t, si conosce quale matrice di intensità $\mathbf{Q}_{X_i \mid pa_i(x)}$, con $pa_i(x) \in Pa(X_i)$, governi la dinamica di X_i.

Perciò la probabilità dei dati $\mathcal{D}$ rispetto a $\mathcal{N}$ è il prodotto delle likelihood di ogni variabile X_i:

$$\begin{aligned} L_{\mathcal{N}}(\mathbf{q}, \theta : \mathcal{D}) &= \prod_{X_i\in\mathbf{X}} L_{X_i}(\mathbf{q}_{X_i \mid Pa(X_i)}, \theta_{X_i \mid Pa(X_i)} : \mathcal{D}) \\ &= \prod_{X_i\in\mathbf{X}} L_{X_i}(\mathbf{q}_{X_i \mid Pa(X_i)} : \mathcal{D})\, L_{X_i}(\theta_{X_i \mid Pa(X_i)} : \mathcal{D}). \end{aligned} \tag{1.8}$$

Il termine $L_X(\theta_{X \mid Pa(X)} : \mathcal{D})$ esprime la likelihood delle transizioni tra stati. Tale termine trascura il tempo che intercorre fra le transizioni poiché esse dipendonono esclusivamente dal valore di nodi genitori (si veda Nodelman *et al.*, 2002, p. 3). Quindi, usando le *statistiche sufficienti* si può scrivere:

$$L_X(\theta_{X \mid Pa(X)} : \mathcal{D}) = \prod_{pa_i(x)} \prod_x \prod_{x\neq x'} (\theta_{xx' \mid pa_i(x)})^{M[x,x' \mid pa_i(x)]}. \tag{1.9}$$

Per quanto riguarda il calcolo di $L_X(\mathbf{q}_{X \mid Pa(X)} : \mathcal{D})$ va considerato il caso in cui il tempo trascorso da X in uno determinato stato x termini non a causa di una sua transizione bensì a causa di una transizione di uno o più nodi appartenenti all'insieme dei suoi nodi genitori (i. e., una nuova istanziazione per l'insieme dei genitori $Pa(X)$). È quindi necessario considerare la probabilità che il nodo X rimanga in x una quantità di tempo *almeno pari* a t mentre i suoi nodi genitori $Pa(X)$ non effettuano alcuna transizione di stato (si veda Nodelman *et al.*,

2002, p. 3). Tale quantità si ricava dalla funzione di distribuzione cumulativa di una distribuzione esponenziale (equazione 1.5):

$$1 - F(t) = e^{-\mathbf{q}_{x|pa_i(x)}t}.$$

Perciò la likelihood delle quantità di tempo trascorse in ogni stato è:

$$L_X(\mathbf{q}_{X|Pa(X)} : \mathcal{D}) = \prod_{pa_i(x)} \prod_{x} (\mathbf{q}_{x|pa_i(x)})^{M[x|pa_i(x)]} e^{-\mathbf{q}_{x|pa_i(x)} T[x|pa_i(x)]}. \tag{1.10}$$

Combinando l'equazione 1.10 e l'equazione 1.9 si ottiene la likelihood di un dataset completo $\mathcal{D}$ rispetto a un singolo processo di Markov condizionale:

$$L_X(\mathbf{q}, \theta : \mathcal{D}) = \prod_{pa_i(x)} \prod_{x} \Bigg[(\mathbf{q}_{x|pa_i(x)})^{M[x|pa_i(x)]} e^{-\mathbf{q}_{x|pa_i(x)} T[x|pa_i(x)]} \cdot \prod_{x \neq x'} (\theta_{xx'|pa_i(x)})^{M[x,x'|pa_i(x)]} \Bigg]. \tag{1.11}$$

Si noti che, dal punto di vista algebrico, è conveniente riformulare l'equazione 1.11 come *log-likelihood*:

$$\ell_X(\mathbf{q}, \theta : \mathcal{D}) = \sum_{pa_i(x)} \sum_{x} \Bigg[M[x|pa_i(x)] \ln(\mathbf{q}_{x|pa_i(x)}) - \mathbf{q}_{x|pa_i(x)} T[x|pa_i(x)] + \sum_{x \neq x'} M[x, x'|pa_i(x)] \ln(\theta_{xx'|pa_i(x)}) \Bigg]. \tag{1.12}$$

È ora possibile asserire che la *log-likelihood* di $\mathcal{N}$ (dall'equazione 1.8) è:

$$\ell_{\mathcal{N}}(\mathbf{q}, \theta : \mathcal{D}) = \sum_{X_i \in \mathbf{X}} \ell_{X_i}(\mathbf{q}, \theta : \mathcal{D}). \tag{1.13}$$

In questa sezione si è presentato come computare la likelihood di un modello di una CTBN rispetto a un dataset completo.

Tuttavia, nel caso in cui non si conoscano i parametri di una CTBN è necessario stimarli. Nella prossima sezione viene affrontato esattamente questo argomento.

1.4.3 Stima dei parametri

Si affronta ora il problema dell'apprendimento dei parametri di una Continuos time Bayesian Network (con struttura nota $\mathcal{G}$) da un insieme di dati completi (si veda Nodelman, 2007, sezione 5.1).

Quando si tratta con dati multinomiali ci sono principalmente due scelte che è possibile fare. La scelta più semplice consiste nell'effettuare una stima dei parametri del modello tramite un approccio *maximum-likelihood*. Tuttavia, è noto che tale approccio può portare a problemi con l'inferenza quando i dati di input sono sparsi. Per evitare tale limitazione solitamente si effettua una *regolarizzazione bayesiana* dei parametri: si sceglie una distribuzione a priori per i parametri e li si aggiorna in accordo ai dati di input.

Come costruire una CIM dai parametri.

La stima dei parametri non è un processo fine a se stesso, in quanto, da essi è possibile costruire le matrici di intensità condizionali (CIM) di ogni nodo della CTBN. Come si ricorderà, una CIM è un insieme di matrici di intensità, una per ogni istanziazione $pa_i(x)$ dei nodi genitori (si veda la definizione 1.10). Perciò, fissato $pa_i(x)$, si può computare la rispettiva matrice di intensità per un nodo qualsiasi ponendo sulla diagonale il rispettivo vettore dei parametri $\mathbf{q}_{x\,|\,pa_i(x)}$ e ricavando i valori non sulla diagonale dalla relazione (si veda il teorema 1.2) fra i parametri $\mathbf{q}$ e θ:

$$\mathbf{q}_{xx'\,|\,pa_i(x)} = \theta_{xx'\,|\,pa_i(x)} \cdot \mathbf{q}_{x\,|\,pa_i(x)}. \tag{1.14}$$

Infine, come vedremo in seguito nel capitolo 3, i parametri sono anche un componente chiave del processo di apprendimento strutturale.

Stima maximum-likelihood

In base a quanto attestato dalla definizione stessa delle CTBN (definizione 1.15 a pagina 12), la dinamica evolutiva globale di una CTBN, cioè la dinamica di tutti i nodi di $\mathcal{G}$ (dei processi di Markov condizionali indicizzati dal tempo), è espressa tramite un processo di Markov omogeneo. Dalla definizione 1.7, inoltre, si deduce che tale processo di Markov induce un modello di probabilità composto da una *distribuzione esponenziale* con parametro $\mathbf{q}_{x\,|\,pa_i(x)}$, che esprime il tempo trascorso in uno stato x da un nodo X data una istanziazione $pa_i(x)$ per i nodi genitori $Pa(X)$, e una *distribuzione multinomiale* con parametro $\theta_{xx'\,|\,pa_i(x)}$, che esprime la probabilità di transizione uscenti dallo stato x verso x' (sempre fermo restando il condizionamento dato dall'istanziazione dei nodi genitori).

La media della distribuzione esponenziale in questione è pari a $1/\mathbf{q}_{x\,|\,pa_i(x)}$. Questa quantità esprime il tempo medio delle transizioni uscenti da uno stato x, fermo restando che il genitore del nodo in questione abbia istanziazione costante e uguale a $pa_i(x)$. Poiché il tempo medio si calcola rapportando il tempo totale trascorso in x, $T[x\,|\,pa_i(x)]$, rispetto al numero totale di transizioni uscenti da x, $M[x\,|\,pa_i(x)]$, si ottiene:

$$\frac{1}{\mathbf{q}_{x\,|\,pa_i(x)}} = \frac{T[x\,|\,pa_i(x)]}{M[x\,|\,pa_i(x)]}.$$

Invece, la probabilità di transizione da uno stato x verso x' sapendo che avverrà una transizione è data dal rapporto tra il numero totale di transizioni da x a x' diviso il numero totale di transizioni uscenti da x; cioè:

$$\frac{M[x, x'\,|\,pa_i(x)]}{M[x\,|\,pa_i(x)]}.$$

Teorema 1.3. *Parametri maximum-likelihood (MLE). I parametri che massimizzano la likelihood (equazione 1.13) di una Continuos time Bayesian Network sono funzione delle statistiche sufficienti:*

$$\begin{aligned} \mathbf{q}_{x \mid pa_i(x)} &= \frac{M[x \mid pa_i(x)]}{T[x \mid pa_i(x)]} \\ \theta_{xx' \mid pa_i(x)} &= \frac{M[x, x' \mid pa_i(x)]}{M[x \mid pa_i(x)]}. \end{aligned} \tag{1.15}$$

Si noti che, in questo caso (*dataset completo*), $\mathbf{q}_{x \mid pa_i(x)}$ e $\theta_{xx' \mid pa_i(x)}$ sono delle *stime esatte*. Essi massimizzano la probabilità a posteriori di un dataset, dato un modello CTBN.

Stima bayesiana

Un approccio alternativo alla stima dei parametri è la stima bayesiana (si veda Nodelman, 2007, sottosezione 5.1.1).

A tal fine è necessario definire una distribuzione a priori sui parametri di una CTBN. Come si è soliti fare in situazioni di questo tipo, per tale distribuzione si sceglie di usare una *distribuzione a priori coniugata*[11] poiché ciò risulta conveniente dal punto di vista algebrico (e quindi computazionale). Infatti, una distribuzione a priori coniugata fornisce un'espressione in forma chiusa per la distribuzione a posteriori (alternativamente potrebbe risultare necessario il calcolo di un integrale numerico).

Si consideri innanzitutto un singolo processo di Markov. Si ricorda (si vedano a tal riguardo le definizioni 1.7 e 1.8 a pagina 7 e a pagina 8) che un processo di Markov ha due insiemi di parametri: θ che parametrizzano una *distribuzione multinomiale* e $\mathbf{q}$ che parametrizzano una *distribuzione esponenziale*.

Una distribuzione a priori coniugata per il parametro $\mathbf{q}$ è la *distribuzione Gamma* $P(\mathbf{q}) = \mathrm{Gamma}(\alpha_x, \tau_x)$, dove (si veda Nodelman, 2007):

$$P(\mathbf{q}) = \frac{\tau_x^{\alpha_{x+1}}}{\Gamma(\alpha_{x+1})} \mathbf{q}^{\alpha_x} e^{-\mathbf{q}\tau_x}. \tag{1.16}$$

Invece, avendo assunto l'indipendenza dei parametri e poiché la funzione di densità della distribuzione di probabilità di θ, che è una multinomiale, è positiva, per essa si sceglie come priori coniugata la *distribuzione di Dirichlet* $P(\theta) = \mathrm{Dir}(\alpha_{xx_1}, \ldots, \alpha_{xx_k})$ (si veda Hecker-

11 In teoria della probabilità bayesiana, se le distribuzioni a posteriori $P(\theta|x)$ sono nella stessa famiglia della distribuzione a priori $P(\theta)$, le due distribuzioni sono definite coniugate, e la distribuzione a priori è chiamata *distribuzione a priori coniugata* per la verosimiglianza (*likelihood*). Una distribuzione a priori coniugata è conveniente dal punto di vista algebrico in quanto fornisce una espressione in forma chiusa per la distribuzione a posteriori e perché può fornire delle intuizioni circa il modo con cui la funzione di verosimiglianza aggiorna la distribuzione.

man, 1996; Heckerman *et al.*, 1995), la cui funzione di densità (Steck, Harald and Jaakkola, 2002) è:

$$P(\theta) = \frac{\Gamma(\alpha_x)}{\Gamma(\alpha_{xx_1}) \cdot \ldots \cdot \Gamma(\alpha_{xx_K})} \theta_{xx_k}^{\alpha_{xx_1}-1} \cdot \ldots \cdot \theta_{xx_1}^{\alpha_{xx_K}-1}. \tag{1.17}$$

Nota 1.4.3.1. Si noti che l'iper-parametro α_x, detto *dimensione equivalente del campione*, è costituito dalla somma dei *conteggi immaginari* $\alpha_{xx_1} + \ldots + \alpha_{xx_K}$, chiamati anche *pseudo-conteggi* (Steck, Harald and Jaakkola, 2002). Esso può essere pensato come un fattore che esprime la *"forza"* della distribuzione a priori, in quanto, più esso aumenta, più le stime dei parametri sono regolarizzate, cioè meno estreme. Chiaramente, quando α_x tende a 0 le stime dei parametri tendono alle stime maximum-likelihood. In letteratura (si veda Steck, Harald and Jaakkola, 2002) questo processo è anche chiamato *"smoothing"*.

Nota 1.4.3.2. L'iper-parametro τ_x, invece, rappresenta una *quantità di tempo immaginaria* che incorpora la credenza della distribuzione a priori sul parametro della distribuzione esponenziale.

Quindi, se si assume che i parametri sono *stocasticamente indipendenti*, cioè che $P(\theta, \mathbf{q}) = P(\theta)\,P(\mathbf{q})$, allora le distribuzioni a posteriori (i. e., condizionate sui dati) dei parametri $\mathbf{q}$ e θ sono:

$$\begin{aligned} P(\mathbf{q} \mid \mathcal{D}) &= \mathrm{Gamma}(\alpha_x + M[x], \tau_x + T[x]) \\ P(\theta \mid \mathcal{D}) &= \mathrm{Dir}(\alpha_{xx_1} + M[x, x_1], \ldots, \alpha_{xx_K} + M[x, x_K]). \end{aligned} \tag{1.18}$$

Al fine di generalizzare quest'idea e ottenere una distribuzione a priori coniugata per un'intera CTBN è necessario che essa soddisfi due assunzioni (comuni per le distribuzioni a priori nelle Bayesian Network, si veda Heckerman (1996)): l'indipendenza globale e locale dei parametri. In base all'*indipendenza globale dei parametri* si può scrivere:

$$P(\mathbf{q}, \theta) = \prod_{X_i \in \mathbf{X}} P(\mathbf{q}_{X_i \mid Pa(X_i)}, \theta_{X_i \mid Pa(X_i)}). \tag{1.19}$$

Invece, dall'*indipendenza locale dei parametri* consegue che è possibile scrivere:

$$P(\mathbf{q}_{X \mid Pa(X)}, \theta_{X \mid Pa(X)}) = \left[\prod_{x} \prod_{pa_i(x)} P(\mathbf{q}_{x \mid pa_i(x)})\right] \left[\prod_{x} \prod_{pa_i(x)} P(\theta_{x \mid pa_i(x)})\right]. \tag{1.20}$$

Se tale distribuzione a priori soddisfa le assunzioni di indipendenza allora anche la distribuzione a posteriori, essendovi coniugata e perciò appartenente alla stessa famiglia parametrica, le soddisferà. In tal caso è possibile mantenere la distribuzione parametrica in forma chiusa e aggiornarla usando le *statistiche sufficienti*:

- $M[x, x' \mid pa_i(x)]$ per il parametro $\theta_{x \mid pa_i(x)}$
- $M[x \mid pa_i(x)]$ e $T[x \mid pa_i(x)]$ per il parametro $\mathbf{q}_{x \mid pa_i(x)}$.

Data una distribuzione sui parametri è possibile usarla per predire il prossimo evento, mediando la sua probabilità sull'insieme dei possibili valori dei parametri. Questo tipo di previsione è equivalente all'utilizzo dei *valori attesi* dei parametri, i quali hanno la stessa forma dei parametri maximum-likelihood ma considerano i *conteggi immaginari* degli iper-parametri:

$$\begin{aligned} \hat{q}_{x|pa_i(x)} &= \frac{\alpha_{x|pa_i(x)} + M[x|pa_i(x)]}{\tau_{x|pa_i(x)} + T[x|pa_i(x)]} \\ \hat{\theta}_{xx'|pa_i(x)} &= \frac{\alpha_{xx'|pa_i(x)} + M[x, x'|pa_i(x)]}{\alpha_{x|pa_i(x)} + M[x|pa_i(x)]}. \end{aligned} \tag{1.21}$$

Si osservi che questi parametri sono, teoricamente, validi solo per predire una singola transizione, dopo la quale la distribuzione dei parametri andrebbe aggiornata di conseguenza. Tuttavia, è prassi comune non aggiornare la distribuzione dei parametri utilizzando i succitati *valori attesi* anche per la previsione delle transizioni successive.

2 CLASSIFICAZIONE

La classificazione è un argomento centrale nei campi di ricerca relativi all'apprendimento automatico (anche detto *machine learning*) e l'analisi dei dati. In generale, essa consiste nel processo di assegnare una *classe* (i. e., un'etichetta) a delle istanze descritte da un insieme di attributi. Si parla di *classificazione supervisionata* quando è necessario indurre un classificatore a partire da un insieme di dati composto da istanze già etichettate e utilizzare tale classificatore per classificare nuove istanze di dati. Nel caso in cui, invece, si voglia individuare dei raggruppamenti intrinseci (i. e., *cluster*) a un insieme di dati composto da istanze non etichettate, e creare in corrispondenza di tali raggruppamenti le classi (incognite), si parla invece di *classificazione non supervisionata*.

In questo capitolo viene quindi introdotta una classe di modelli, che prende il nome di Continuos time Bayesian Network classifier (CTBNC), il cui scopo è la *classificazione supervisionata* di traiettorie multivariate di variabili discrete a *tempo continuo*. Si descrivono due istanze di tale classe: i classificatori Continuos time Naive Bayes (CTNB) e i classificatori Continuos time tree augumented Naive Bayes (CTTANB).

Mentre nella sezione 2.2 si affronta il processo di *apprendimento* in caso di *dati completi* dei CTBNC, nella sezione 2.3, si presenta un algoritmo di *inferenza esatta* per la classe dei CTBNC.

2.1 MODELLO

Al fine di risolvere il succitato problema della classificazione sono stati proposti numerosi approcci. Ad esempio naive Bayes classifier, un classificatore semplice ma robusto proposto da Duda e Hart (1973); rivelatosi essere uno fra i classificatori più performanti (Langley *et al.*, 1992). Esso apprende dai dati la probabilità condizionale di ogni attributo A_i data la classe C. La classificazione di nuove istanze dei dati è effettuata applicando la *regola di Bayes* al fine di calcolare la probabilità della classe C data l'istanziazione di $A_i, \ldots, A_N$ e scegliendo quella con la maggiore probabilità a posteriori. Questo calcolo è reso possibile in modo efficiente grazie ad una forte assunzione: tutti gli attributi A_i sono *condizionalmente indipendenti* (si veda la definizione 1.2) tra di loro data evidenza sulla classe C.

Poiché tale assunzione è chiaramente irreale, Friedman *et al.* (1997) ha investigato come migliorare ulteriormente le prestazioni del naive

Bayes classifier evitando assunzioni di indipendenza non giustificate dai dati. A tal fine Friedman *et al.* (1997), generalizzando il naive Bayes classifier, ha proposto una classe di modelli di *classificazione supervisionata*, chiamata Bayesian Network classifier (BNC) (di cui fa parte il Tree Augumented Naive Bayes (TAN) classifier, ad esempio) che ereditano dalla teoria delle Bayesian Network (si rimanda alla definizione 1.1 per maggiori dettagli) una rappresentazione fattorizzata delle distribuzioni di probabilità dei nodi attributo e rappresentano esplicitamente le indipendenze condizionali fra essi.

Seguendo le stesse motivazioni, in Stella e Amer (2012) viene formalizzata una classe di modelli di *classificazione supervisionata*, chiamati Continuos time Bayesian Network classifier (CTBNC), derivata dalle CTBN (si veda definizione 1.15).

Di seguito si definiscono quindi i Continuos time Bayesian Network classifier e due istanze di classificatori appartenenti a tale classe: il Continuos time Naive Bayes classifier (CTNBC) e il Continuos time tree augumented Naive Bayes classifier (CTTANBC).

Un Continuos time Bayesian Network classifier estende una CTBN tramite l'aggiunta di un nodo associato alla variabile classe Y. Si ricorda, dalla definizione 1.15, che una CTBN rappresenta l'evoluzione nel tempo continuo di una variabile di processo $\mathbf{X}$ (i. e., insieme composto da N processi di Markov, si veda la definizione 1.11).

Di seguito si dà la definizione di questa nuova classe di modelli di *classificazione supervisionata*.

Definizione 2.1 (Continuos time Bayesian Network classifier)**.** Un Continuos time Bayesian Network classifier (CTBNC) è composto da una coppia $\mathcal{C} = (\mathcal{N}, \mathbf{P}(Y))$ dove:

- $\mathcal{N}$ è una CTBN con nodi attributo $X_1, X_2, \ldots, X_N$
- Y è il nodo classe con valori $val(Y) = \{y_1, \ldots, y_K\}$ e probabilità marginale $\mathbf{P}(Y)$.

E inoltre il grafo su $\mathcal{N}$ (i. e., il grafo $\mathcal{G}$, si veda la definizione 1.15) rispetta le seguenti condizioni:

- $\mathcal{G}$ è un grafo connesso[12]
- $Pa(Y) = \{\}$, i. e., la variabile casuale Y è associata al nodo classe
- il nodo Y è indipendente dal tempo ed è specificato solo ed esclusivamente dalla sua probabilità marginale $\mathbf{P}(Y)$.

A supporto della definizione 2.1, la figura 2.1 nella pagina seguente fornisce un'istanza di CTBNC composta dai nodi attributi X_1, X_2, X_3, X_4, X_5 e dal nodo classe Y (nodo radice). Si osservi come tale istanza contenga dei cicli, uno riguardante i nodi X_2, X_4, X_5, X_3 e l'altro riguardante

12 Il grafo $\mathcal{G} = (V, E)$ è detto *connesso* se $\forall (u, v) \in V$ esiste un cammino che collega u a v.

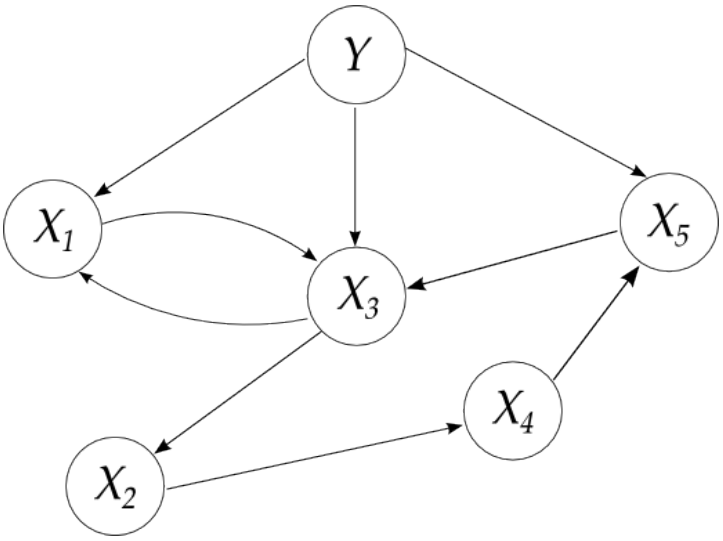


Figura 2.1: Un esempio di Continuos time Bayesian Network classifier (CTBNC) con cinque nodi attributo, $X_1, \ldots, X_5$, e un nodo classe, Y.

i nodi X_1, X_3. Si fa notare che gli archi della rete $\mathcal{N}$ rappresentano le dipendenze causali nel tempo.

Parallelamente a quanto fatto in Langley *et al.* (1992), si presentano ora due istanze particolari di Continuos time Bayesian Network classifier.

Definizione 2.2 (Continuos time Naive Bayes classifier)**.** Un Continuos time Naive Bayes classifier (CTNBC) è un Continuos time Bayesian Network classifier $\mathcal{C} = (\mathcal{N}, \mathbf{P}(Y))$ caratterizzato dal fatto che ogni nodo attributo ha un solo genitore, il nodo classe Y. Risulta quindi che:

$$\mathrm{Pa}(X_i) = \{Y\} \quad \forall\, X_i \in \mathcal{G}.$$

Come mostrato dalla figura 2.2, un CTNBC possiede un nodo radice, associato alla variabile casuale Y, che è l'unico genitore di tutti i restanti nodi X_i (con $i = 1, 2, \ldots, N$) che lo compongono. Si osservi come la rete di un CTNBC rappresenti l'assunzione di *indipendenza condizionale* di ogni nodo attributo dagli altri, data evidenza sulla variabile classe Y.

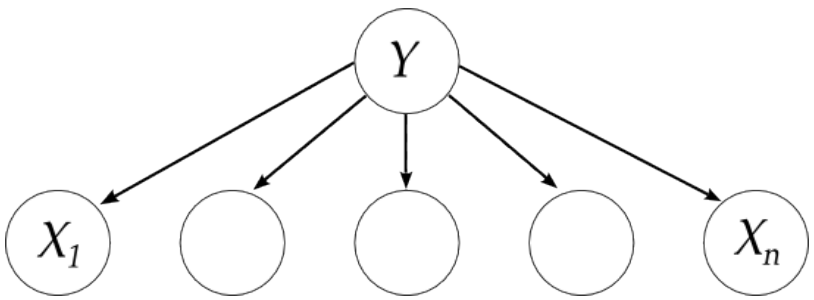


Figura 2.2: Un Continuos time Naive Bayes classifier (CTNBC).

Definizione 2.3 (Continuos time tree augumented Naive Bayes classifier)**.** Un Continuos time tree augumented Naive Bayes classifier (CTTANBC) è un Continuos time Bayesian Network classifier $\mathcal{C} = (\mathcal{N}, \mathbf{P}(Y))$ che rispetta i seguenti vincoli:

- $Y \in Pa(X_i)$ con $i = 1, 2, \ldots, N$
- i nodi attributo X_i, $i = 1, 2, \ldots, N$, formano un albero:

$$\exists\, j \;:\quad |Pa(X_j)| = 1 \quad \text{mentre per} \quad i \neq j \;:\quad |Pa(X_i)| = 2.$$

Come mostrato dalla figura 2.3, un classificatore CTTANB è un estensione del classificatore CTNB: tutti i nodi attributo della rete $\mathcal{N}$ sono vincolati ad avere come genitore, oltre al nodo radice, al massimo un altro nodo attributo. Ciò comporta che tutti i nodi attributo facciano parte del *Markov blanket* del nodo radice associato con la variabile classe Y.

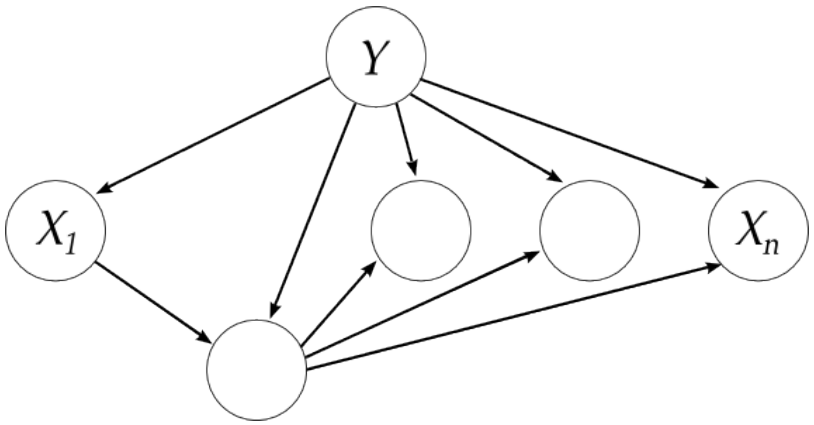


Figura 2.3: Un Continuos time tree augumented Naive Bayes classifier (CTTANBC): qualora la variabile classe Y venga rimossa, le variabili rimanenti formano un albero.

2.2 APPRENDIMENTO

In questa sezione si affronta il problema dell'apprendimento (da *dati completi*) dei CTBNC.

Per definizione (si veda la definizione 2.1 a pagina 22) i CTBNC sono basati sul modello delle CTBN, rappresentano perciò un insieme di modelli di probabilità locali (relativi alle variabili casuali) esprimibili in termini di statistiche sufficienti (per maggiori dettagli relativi a questo aspetto si rimanda alla sezione 1.4). Ne deriva che il problema dell'apprendimento di un classificatore CTBN si riduce alla computazione delle statistiche sufficienti dei suoi nodi attributo, da cui è successivamente possibile stimare i parametri (argomento trattato in dettaglio nella sottosezione 1.4.3) delle distribuzioni di probabilità codificate dalle matrici di intensità condizionali (CIM).

Di conseguenza, per l'apprendimento di un CTNBC è richiesto uno sforzo computazionale minimo. Per l'apprendimento di un CTTANBC, poiché questo modello prevede archi anche fra i nodi attributo, è invece richiesto uno sforzo computazionale leggermente maggiore (Stella e Amer, 2012).

Si presenta di seguito l'algoritmo 2.1 (Stella e Amer, 2012) relativo all'apprendimento di un classificatore CTNB (definizione 2.2).

Esso richiede in input un *dataset completo* $\mathcal{D} = \{\delta_1, \ldots, \delta_h\}$, il corrispettivo insieme delle classi $\{y_1, y_2, \ldots, y_h\}$, con $y_i \in val(Y)$, e il grafo $\mathcal{G}$ di una CTBN $\mathcal{N}$ (rispettivamente chiamati `data`, `classes` e `graph` nella firma della funzione `ctnbclearn`).

Per completezza si osservi (in base alla definizione 1.16) che ogni δ_i (con $i = 1, \ldots, h$) è un flusso di evidenze $(\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^{J_i})$ (a tal riguardo si veda la definizione 1.14).

Il risultato dell'applicazione dell'algoritmo 2.1 è un Continuos time Naive Bayes classifier $\mathcal{C} = (\mathcal{N}, \mathbf{P}(Y))$.

```
1  function ctnbclearn(data, classes, graph) {
2      var h = len(data)
3      var klass = unique(classes)
4      var nk = len(class)
5      var priors[nk]
6      for (i in keyword(data)) {
7          var k = keyword(classes[i])
8          priors[k] = priors[k] + (1 / h)
9      }
10     var m(), t()
11     for (i in keyword(data)) {
12         var y = classes[i]
13         var j = 1
14         while (t_j ⩽ T_i) {
15             for (n in keyword(graph.nodes)) {
16                 m(x_n^j, x_n^{j+1}, y) = m(x_n^j, x_n^{j+1}, y) + 1
17                 t(x_n^j, y) = t(x_n^j, y) + (t_j − t_{j−1})
18             }
19             j = j + 1
20         }
21     }
22     var q(), thet()
23     foreach (y in klass) {
24         for (n in keyword(graph.nodes)) {
25             for (x_n in val(X_n)) {
26                 var mm(x_n, y) = ∑_{x'_n ≠ x_n} m(x_n, x'_n, y)
27                 q(x_n, y) = mm(x_n, y) / t(x_n, y)
28                 thet(x_n, y) = m(x_n, x'_n, y) / mm(x_n, y)
29             }
```

```
30          }
31      }
32      var ctbn = new ctbn(graph, q, thet)
33      return (priors, ctbn)
34  }
```

Algoritmo 2.1: Apprendimento di un classificatore CTNB

L'algoritmo di apprendimento appena presentato consiste nella stima delle matrici di intensità condizionali (CIM) di ogni variabile casuale di $\mathcal{N}$ per ogni classe $y_i \in Y$. Più in dettaglio, esso è composto da tre fasi consecutive:

1. da linea 2 a linea 9 viene calcolata la *probabilità a priori* della variabile classe Y in base alla frequenza di ogni sua istanziazione $y_i \in Y$ in $\mathcal{D}$
2. da linea 10 a linea 21 vengono calcolate le statistiche sufficienti di ogni nodo attributo X_i, con $i = 1, 2, \ldots, N$, sull'insieme di dati di apprendimento (anche detto *training set*) $\mathcal{D}$
3. da linea 22 a linea 31 vengono infine stimati, a partire dalle statistiche sufficienti, i *parametri maximum-likelihood* (MLE).

Si osservi che, poiché il processo di apprendimento è eseguito su un classificatore CTNB, l'algoritmo condiziona sia il calcolo delle statistiche sufficienti (i. e., variabili `t` e `m`) che la stima dei parametri (i. e., variabili `q` e `thet`) di ogni nodo attributo X_i solo ed esclusivamente al valore della variabile classe Y (i. e., variabile `y`). Questa semplificazione è dovuta al vincolo che caratterizza i classificatori CTNB: ogni nodo attributo X_i ha un solo genitore, il nodo associato alla variabile classe Y (si veda la definizione 2.2).

Come anticipato, al costo di un leggero incremento di complessità computazionale, è possibile estendere l'algoritmo 2.1 al fine di creare un algoritmo di apprendimento generale che apprenda un qualsiasi classificatore CTBN. Affinché tale obiettivo sia raggiunto è necessario rimuovere il succitato vincolo sull'insieme dei genitori di ogni nodo attributo. Mentre il calcolo della probabilità a priori della variabile classe Y non varia, il calcolo delle statistiche sufficienti e la stima dei parametri, invece, necessitano di tale generalizzazione.

Nello specifico:

1. il calcolo delle statistiche sufficienti di ogni nodo attributo X_i va condizionato all'istanziazione attuale (i. e., al tempo `j`) del suo insieme di nodi genitori $\mathrm{Pa}(X_i)$; perciò a linea 15 dell'algoritmo 2.2 si prende in considerazione tale valore (i. e., variabile `p`)
2. la stima dei parametri maximum-likelihood (MLE) di ogni noto attributo X_i va eseguita in base a ogni istanziazione del suo insieme di nodi genitori $\mathrm{Pa}(X_i)$; perciò a linea 25 si itera in

base a ogni valore (i. e., variabile p) assunto da $Pa(X_i)$ (i. e., $val(Pa(X_i))$) mentre l'iterazione per classe non è più coerente e di conseguenza rimossa.

Si riporta di seguito l'algoritmo 2.2, il quale include le succitate modifiche finalizzate alla creazione di un algoritmo di apprendimento generale per i CTBNC.

```
function learn(data, classes, graph) {
    var h = len(data)
    var klass = unique(classes)
    var nk = len(class)
    var priors[nk]
    for (i in keyword(data)) {
        var k = keyword(classes[i])
        priors[k] = priors[k] + (1 / h)
    }
    var m(), t()
    foreach (i in keyword(data)) {
        var j = 1
        while (t_j ⩽ T_i) {
            foreach (n in keyword(graph.nodes)) {
                var p = val^j(Pa(X_n))
                m(x_n^j, x_n^{j+1}, p) = m(x_n^j, x_n^{j+1}, p) + 1
                t(x_n^j, p) = t(x_n^j, p) + (t_j − t_{j−1})
            }
            j = j + 1
        }
    }
    var q(), thet()
    foreach (n in keyword(graph.nodes)) {
        for (x_n in val(X_n)) {
            foreach (p in val(Pa(X_n))) {
                var mm(x_n, p) = ∑_{x'_n ≠ x_n} m(x_n, x'_n, p)
                q(x_n, p) = mm(x_n, p) / t(x_n, p)
                thet(x_n, p) = m(x_n, x'_n, p) / mm(x_n, p)
            }
        }
    }
    var ctbn = new ctbn(graph, q, thet)
    return (priors, ctbn)
}
```

Algoritmo 2.2: Apprendimento di un classificatore CTBN

2.3 INFERENZA

In questa sezione si affronta il problema della *classificazione* di un *flusso di evidenze completamente osservato*, indicato con $(\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^J)$ (si veda la definizione 1.14), rispetto a un classificatore CTBN (CTBNC). Si osservi che, in tale situazione (i. e., *dati completi*), l'unica variabile casuale non osservata è la variabile classe, perciò è possibile sfruttare le relazioni di indipendenza fra variabili casuali così come si fa per le Bayesian Network. L'argomento di questa sezione è quindi il processo di *classificazione supervisionata*, di cui si presentano in primis le basi teoriche e successivamente l'implementazione algoritmica che ne consegue.

Il processo di classificazione di un flusso di evidenze è effettuato in base alla regola *maximum a posteriori*[13] (MAP) (si veda Stella e Amer, 2012): un flusso di evidenze completamente osservato viene classificato assegnandogli la classe la cui probabilità a posteriori (rispetto al flusso di evidenze stesso) è massima. A tale scopo è necessario calcolare la probabilità a posteriori della variabile classe Y del CTBNC, rispetto al flusso di evidenze in input, per tutti i suoi possibili stati (i. e., classi, o etichette).

Il classificatore CTBN classifica quindi il flusso di evidenze massimizzando la seguente probabilità a posteriori, ricavata applicando la *regola di Bayes*:

$$P(Y \mid (\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^J)) = \frac{P((\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^J) \mid Y)\, P(Y)}{P((\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^J))}. \tag{2.1}$$

Si specifica di seguito la semantica dei componenti dell'equazione 2.1:

- la probabilità marginale associata alla variabile classe Y

$$P(Y)$$

- la probabilità del flusso di evidenze

$$P((\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^J))$$

- la likelihood del flusso di evidenze dato il valore della variabile classe, a cui ci si riferisce nel prosieguo usando l'espressione *"likelihood temporale"*

$$P((\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^J) \mid Y).$$

13 La stima della probabilità maximum a posteriori (MAP) è una moda della distribuzione a posteriori che può essere usata per ottenere una stima puntuale di una quantità inosservata sulla base di dati empirici. Può essere vista come una regolarizzazione della stima maximum-likelihood (MLE) poiché è strettamente correlata ad essa; vi differisce perché impiega un obiettivo di massimizzazione incrementato che incorpora una distribuzione a priori sopra la quantità che si vuole stimare.

La probabilità del flusso di evidenze (i. e., denominatore dell' equazione 2.1), similmente a quanto accade per i BNC (Friedman *et al.*, 1997), può essere omessa poiché sussiste la relazione di proporzionalità tra il numeratore e la probabilità che si intende calcolare:

$$P(Y \mid (\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^J)) \propto P((\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^J) \mid Y)\, P(Y). \tag{2.2}$$

Il primo termine dell'equazione 2.2, cioè la *likelihood temporale*, è invece fondamentale per la classificazione tramite regola MAP ed è possibile riformularlo nel seguente modo:

$$P((\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^J) \mid Y) = \prod_{j=1}^{J} P(\mathbf{x}^j \mid Y)\, P(\mathbf{x}^{j+1} \mid \mathbf{x}^j, Y), \tag{2.3}$$

dove:

- $P(\mathbf{x}^j \mid Y)$ rappresenta la probabilità che il *vettore aleatorio*[14] $\mathbf{X}$ resti nello stato $\mathbf{x}^j$ durante l'intervallo temporale $[t_{j-1}, t_j)$ data evidenza sulla variabile classe Y
- $P(\mathbf{x}^{j+1} \mid \mathbf{x}^j, Y)$ rappresenta la probabilità che in $\mathbf{X}$ si verifichi una transizione da $\mathbf{x}^j$ a $\mathbf{x}^{j+1}$ all'istante di tempo t_j data evidenza sulla variabile classe Y.

Inoltre, al fine di assicurare la consistenza dell'equazione 2.3 si assume che $P(\mathbf{x}^{J+1} \mid \mathbf{x}^J, Y) = 1$.

Il passo successivo consiste nel calcolo dei due termini da cui è composta l'equazione 2.3. A tal fine si utilizzano le distribuzioni di probabilità locali associate ad ogni nodo del CTBN su cui è costruito il classificatore. Come già descritto nella sottosezione 1.1.2, tali modelli di probabilità sono espressi tramite le matrici di intensità condizionali e quindi tramite i parametri $\mathbf{q}$ e θ.

In tale contesto è quindi possibile calcolare il termine $P(\mathbf{x}^j \mid Y)$ come segue:

$$P(\mathbf{x}^j \mid Y) = \prod_{n=1}^{N} \exp\Big(- \mathbf{q}_{x_n^j}^{pa_j(x_n)} (t_j - t_{j-1}) \Big), \tag{2.4}$$

dove $\mathbf{q}_{x_n^j}^{pa_j(x_n)}$ è il valore del parametro della *distribuzione esponenziale* quando la variabile casuale X_n è nello stato x_n durante il j-esimo intervallo temporale $[t_{j-1}, t_j)$ e contemporaneamente l'istanziazione dei genitori di X_n è $pa_j(x_n)$.

Ugualmente, il termine $P(\mathbf{x}^{j+1} \mid \mathbf{x}^j, Y)$ è così calcolabile:

$$P(\mathbf{x}^{j+1} \mid \mathbf{x}^j, Y) = \prod_{n=1}^{N} P(x_n^{j+1} \mid x_n^j, Y), \tag{2.5}$$

14 Un *vettore aleatorio* $\mathbf{X} = (X_1, X_2, \dots, X_n)$ è una n-*upla* (composta da n variabili casuali) i cui elementi sono dati da numeri aleatori.

dove

$$P(x_n^{j+1} \mid x_n^j, Y) = \begin{cases} \mathbf{q}_{x_n^j x_n^{j+1}}^{pa_j(x)}, & \text{se } x_n^j \neq x_n^{j+1} \\ 1, & \text{altrimenti} \end{cases}. \tag{2.6}$$

Il termine $P(x_n^{j+1} \mid x_n^j, Y)$ rappresenta la probabilità che nel vettore aleatorio $\mathbf{X}$ si verifichi una transizione dallo stato $\mathbf{x}^j$ allo $\mathbf{x}^{j+1}$, dato il valore della variabile classe Y.

Poiché il modello CTBN implica che, ad ogni istante t_j, solo un componente X_n del vettore aleatorio $\mathbf{X}$ può essere soggetto a transizione, allora $P(x_n^{j+1} \mid x_n^j, Y)$ rappresenta la probabilità che la variabile casuale X_n effettui una transizione da x_n^j a x_n^{j+1} mentre tutti gli altri componenti del vettore aleatorio $\mathbf{X}$ (i. e., X_i con $i \neq n$) non cambiano il proprio stato.

Perciò, come specificato dall'equazione 2.6, nel caso in cui avvenga un cambio di stato in X_n, il termine $P(x_n^{j+1} \mid x_n^j, Y)$ equivale alla quantità $\mathbf{q}_{x_n^j x_n^{j+1}}^{pa_j(x)}$, ricavata dalla relazione fra i parametri (si veda il teorema 1.2). Tale quantità rappresenta il parametro associato alla transizione da x_n^j, stato in cui la variabile casuale X_n si trovava durante il j-esimo intervallo temporale $[t_{j-1}, t_j)$, a x_n^{j+1}, stato in cui X_n si troverà durante il successivo intervallo temporale $[t_j, t_{j+1})$; data l'istanziazione $pa_j(x_n)$ dei genitori di X_n durante il j-esimo intervallo temporale.

Combinando l'equazione 2.4 e l'equazione 2.5 si ottiene:

$$P(\mathbf{x}^j \mid Y)\, P(\mathbf{x}^{j+1} \mid \mathbf{x}^j, Y) = \prod_{n=1}^{N} \exp\Big(-\mathbf{q}_{x_n^j}^{pa_j(x_n)} (t_j - t_{j-1}) \Big)\, P(x_n^{j+1} \mid x_n^j, Y). \tag{2.7}$$

Utilizzando l'equazione 2.7 appena ricavata è possibile riformulare la *likelihood temporale* (equazione 2.3) nel seguente modo:

$$P((\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^J) \mid Y) = \prod_{j=1}^{J} \prod_{n=1}^{N} \exp\Big(-\mathbf{q}_{x_n^j}^{pa_j(x_n)} (t_j - t_{j-1}) \Big)\, P(x_n^{j+1} \mid x_n^j, Y). \tag{2.8}$$

Sostituendo infine l'equazione equazione 2.8 nell'equazione 2.2 si formula definitivamente la probabilità a posteriori della variabile classe Y dato un flusso di evidenze:

$$P(Y \mid (\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^J)) \propto P(Y) \cdot \prod_{j=1}^{J} \prod_{n=1}^{N} \Bigg[\exp\Big(-\mathbf{q}_{x_n^j}^{pa_j(x_n)} (t_j - t_{j-1}) \Big) \cdot \cdot P(x_n^{j+1} \mid x_n^j, Y) \Bigg]. \tag{2.9}$$

Di conseguenza, dato un classificatore CTBN $\mathcal{C} = \{\mathcal{N}, P(Y)\}$ e un flusso di evidenze completamente osservato $(\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^J)$, la regola MAP seleziona la classe $y^* \in val(Y)$ massimizzando l'equazione 2.9:

$$y^* = \arg\max_{y \in val(Y)} P(Y) \prod_{j=1}^{J} \prod_{n=1}^{N} \exp\Big(-\mathbf{q}_{x_n^j}^{pa_j(x_n)} (t_j - t_{j-1}) \Big)\, P(x_n^{j+1} \mid x_n^j, Y) \tag{2.10}$$

Si presenta di seguito l'algoritmo per l'*inferenza esatta* (Stella e Amer, 2012) di un flusso di evidenze completamente osservato rispetto a un classificatore CTBN.

Tuttavia si osservi che tale algoritmo rappresenta le probabilità come *log-probabilità*[15], a causa della succitata convenienza algebrica che ne deriva. Ciò significa che l'algoritmo 2.3 implementa la probabilità a posteriori della classe dato un flusso di evidenze (equazione 2.9) come segue:

$$\ell_{P(Y|\ldots)} = \log(P(Y)) + \sum_{j=1}^{J} \sum_{n=1}^{N} -\mathbf{q}_{x_n^j}^{pa_j(x_n)} (t_j - t_{j-1}) + \log(P(x_n^{j+1} \mid x_n^j, Y)). \tag{2.11}$$

A tal proposito si noti che, nel caso in cui non avvenga alcun cambio di stato durante un determinato istante di tempo t_j, la quantità $P(x_n^{j+1} \mid x_n^j, Y)$ dell'equazione 2.11 sarà pari a 1 (si veda l'equazione 2.6), il cui logaritmo è pari a 0. Ciò si riflette nella struttura di controllo condizionale alla linea 11.

```
1  function infer(ctbnc, timeseg, stream) {
2      var priors = ctbnc.priors
3      var logp[len(priors)]
4      for (k in keyword(priors)) {
5          logp[k] = log(priors[k])
6      }
7      for (k in keyword(priors)) {
8          for (j in keyword(timeseg)) {
9              for (n in keyword(ctbnc.graph.nodes)) {
10                 logp[k] = logp[k] - q_{x_n^j}^{pa_j(x_n)} * timeseg[j]
11                 if (x_{n_j} != x_{n_{j+1}}) {
12                     logp[k] = logp[k] + log(q_{x_n^j x_n^{j+1}}^{pa_j(x)})
13                 }
14             }
15         }
16     }
17     return which(max(logp))
18 }
```

Algoritmo 2.3: Inferenza su un classificatore CTBN

L'algoritmo 2.3 di inferenza esatta appena presentato è composto principalmente da due fasi corrispondenti ai due termini principali

15 La *log-probabilità* è un modalità di rappresentazione della probabilità che porta con sè alcuni vantaggi algebrici e computazionali. Ad esempio, l'utilizzo della *log-probabilità* generalmente comporta una maggior velocità dovuta alla trasformazione delle moltiplicazioni, più computazionalmente costose, in addizioni. In informatica è molto comune l'utilizzo della sua variante negativa, la quale codifica un valore di probabilità $x \in [0, 1]$ come $x' = -\log(x) \in \mathbb{R}$.

dell'equazione 2.11: da linea 2 a linea 6 converte la distribuzione della variabile classe Y del classificatore CTBN in forma logaritmica; da linea 7 a linea 16 aggiorna, incrementandola o decrementandola, il valore di *log-probabilità* relativo a ogni classe (si veda il ciclo **`for`** alla linea 7) del classificatore CTBN iterando il flusso di evidenze per ogni segmento temporale cui sono associate le sue istanze (i. e., variabile `timeseg`, che si assume venga fornita in input; si veda il ciclo **`for`** alla linea 8), infine iterando sui nodi del grafo $\mathcal{N}$ del CTBNC (ciclo **`for`** a linea 9).

Nota 2.3.1. Si osservi che è possibile implementare l'algoritmo di inferenza anche utilizzando la *parametrizzazione mista* (si veda la definizione 1.8). In tal caso si evita la computazione delle matrici di intensità condizionali (CIM) ma, in contrasto, è necessario calcolare il termine $\mathbf{q}^{\mathrm{pa}_j(x)}_{x_n^j x_n^{j+1}}$ (a tal proposito si veda l'equazione 1.14) ogni qual volta una variabile casuale del flusso di evidenze di input effettui una transizione (a patto che tale transizione avvenga fra due stati appartenenti allo spazio degli stati della rispettiva variabile casuale associata al nodo del classificatore CTBN).

Il passo finale dell'algoritmo, corrispondente alla linea linea 17, consiste nella restituzione della classe la cui *log-probabilità* è maggiore di quella delle restanti classi. Tale passo completa perciò l'implementazione dell'equazione 2.10.

L'algoritmo di inferenza esatta presentato in questa sezione è polinomiale.

3 APPRENDIMENTO STRUTTURALE

Uno dei casi principali che costituisce il problema dell'*apprendimento* di modelli grafico probabilistici è l'apprendimento della struttura incognita sottostante un modello, cioè la selezione di un modello probabilistico che rappresenti un dato *training set*. Anche se questo processo è spesso di elevata complessità, esso è considerato di elevata importanza in quanto la costruzione manuale di una struttura può essere difficile o addirittura impossibile se le variabili che la compongono sono sconosciute o di difficile caratterizzazione.

Il problema dell'*apprendimento strutturale* da *dati completi* di una Continuos time Bayesian Network (CTBN) è quindi l'argomento trattato in questo capitolo.

Generalmente, questo problema può essere informalmente descritto nel seguente modo: dato un *training set* composto da istanze di un insieme di variabili casuali si trovi un grafo che rappresenti tali dati e le relazioni fra le variabili casuali. Si noti come tale problema possa essere catalogato come una forma di apprendimento non supervisionato; nel senso che il processo di apprendimento non distingue la variabile classe dalle variabili attributo nei dati. L'obiettivo è quindi indurre una struttura (i. e., grafo) che descriva nel miglior modo possibile la distribuzione di probabilità sui dati (i. e., *training set*). Si osservi, inoltre, che questo problema di ottimizzazione è solitamente intrattabile per le Bayesian Network (Chickering, 1994; Chickering *et al.*, 2004), anche qualora si restringa la cardinalità massima dell'insieme dei genitori di ogni nodo.

Per quanto riguarda invece il caso delle CTBN, Nodelman *et al.* (2002) hanno dimostrato che, grazie alla mancanza del vincolo di aciclicità, come già accennato nella sezione 1.3, il problema dell'apprendimento strutturale di una CTBN è significativamente più facile, sia teoricamente che in pratica, rispetto all'apprendimento strutturale di una Bayesian Network, o di modelli da esse derivanti (e. g., le Dynamic Bayesian Networks (DBN)). Inoltre, nel caso si vincoli la procedura di ricerca a strutture con un numero massimo di genitori per nodo, questo problema può essere risolto in tempo polinomiale.

L'approccio che si presenta in questo capitolo è quindi un approccio basato sul punteggio: si definisce una funzione che computa uno *score bayesiano* finalizzato alla valutazione di ogni struttura rispetto ai dati di addestramento (i. e., *training set*) e si usa una tecnica di ricerca euristica (e. g., la ricerca *hill climbing*) per cercare nello spazio delle strutture candidate quella che esibisce il maggior punteggio.

Si osservi che l'apprendimento dei parametri (si veda la sottosezio-

ne 1.4.3) è propedeutico per tale obiettivo poiché essi costituiscono la base dello score bayesiano.

3.1 FUNZIONE DI SCORING

Qualsiasi processo di apprendimento strutturale basato su punteggio è costituito da due componenti: una *funzione di scoring* e una procedura di ottimizzazione.

L'obiettivo di questa sezione è quindi presentare una funzione di scoring per l'apprendimento strutturale delle Continuos time Bayesian Network (CTBN). Lo scopo di tale funzione è calcolare il punteggio (i. e., lo score bayesiano) di una struttura relativamente al *training set* $\mathcal{D}$ fornito.

Si definisce lo *score bayesiano* sul grafo $\mathcal{G}$ di una CTBN nel seguente modo:

$$\text{score}_B(\mathcal{G} : \mathcal{D}) = \ln P(\mathcal{D} \mid \mathcal{G}) + \ln P(\mathcal{G}) \tag{3.1}$$

Come mostra l'equazione 3.1 la funzione di scoring utilizza la probabilità a posteriori dell'insieme dei dati di apprendimento (i. e., il training set $\mathcal{D}$) data la struttura candidata (i. e., $\mathcal{G}$), oltre alla probabilità a priori della struttura stessa.

È possibile aumentare in modo significativo l'efficienza dell'algoritmo di ricerca che si affronta nella prossima sezione qualora si facciano determinate assunzioni. Nello specifico, se si assume che la probabilità a priori della struttura, $P(\mathcal{G})$, soddisfi la *modularità della struttura*, ne consegue:

$$P(\mathcal{G}) = \prod_{X_i} P(Pa(X_i) = Pa_{\mathcal{G}}(X_i)). \tag{3.2}$$

Se si assume, inoltre, che la probabilità a priori dei parametri soddisfi la *modularità dei parametri*, allora per ogni due strutture $\mathcal{G}$ e $\mathcal{G}'$ tali che $Pa_{\mathcal{G}}(X) = Pa_{\mathcal{G}'}(X)$ risulta:

$$P(\mathbf{q}_X, \theta_X \mid \mathcal{G}) = P(\mathbf{q}_X, \theta_X \mid \mathcal{G}'). \tag{3.3}$$

Combinando l'assunzione di *indipendenza dei parametri* con l'equazione 3.3 derivante dalla *modularità dei parametri*, si ottiene:

$$P(\mathbf{q}_{\mathcal{G}}, \theta_{\mathcal{G}} \mid \mathcal{G}) = \prod_{X_i} \left[P\left(\mathbf{q}_{X_i \mid Pa(X_i)} \mid Pa(X_i) = Pa_{\mathcal{G}}(X_i)\right) \cdot \right.$$
$$\left. \cdot P\left(\theta_{X_i \mid Pa(X_i)} \mid Pa(X_i) = Pa_{\mathcal{G}}(X_i)\right) \right]. \tag{3.4}$$

Si osservi che, poiché la *penalità del grafo*, corrispondente al termine $P(Pa(X_i) = Pa_{\mathcal{G}}(X_i))$ dell'equazione 3.2, è legata alla dimensione del grafo ma indipendente dalla quantità dei dati, è possibile ignorare il termine $P(\mathcal{G})$ della funzione di scoring (equazione 3.1).

Di conseguenza l'unico termine significativo dell'equazione 3.1 è la *likelihood marginale*, $P(\mathcal{D} \mid \mathcal{G})$. Tale termine, infatti, codifica l'incertezza sui parametri integrando su tutti i possibili valori che essi possono assumere:

$$P(\mathcal{D} \mid \mathcal{G}) = \int_{\mathbf{q}_\mathcal{G}, \boldsymbol{\theta}_\mathcal{G}} P(\mathcal{D} \mid \mathbf{q}_\mathcal{G}, \boldsymbol{\theta}_\mathcal{G})\, P(\mathbf{q}_\mathcal{G}, \boldsymbol{\theta}_\mathcal{G} \mid \mathcal{G})\, d\mathbf{q}_\mathcal{G} d\boldsymbol{\theta}_\mathcal{G}. \tag{3.5}$$

Come per l'equazione 1.8, la likelihood marginale può essere decomposta come un prodotto di likelihood:

$$\begin{aligned} P(\mathcal{D} \mid \mathbf{q}_\mathcal{G}, \boldsymbol{\theta}_\mathcal{G}) &= \prod_{X_i} L_{X_i}(\mathbf{q}_{X_i \mid Pa(X_i)} : \mathcal{D})\, L_{X_i}(\boldsymbol{\theta}_{X_i \mid Pa(X_i)} : \mathcal{D}) \\ &= \underbrace{\left[\prod_{X_i} L_{X_i}(\mathbf{q}_{X_i \mid Pa(X_i)} : \mathcal{D})\right]}_{L(\mathbf{q}:\mathcal{D})} \underbrace{\left[\prod_{X_i} L_{X_i}(\boldsymbol{\theta}_{X_i \mid Pa(X_i)} : \mathcal{D})\right]}_{L(\boldsymbol{\theta}:\mathcal{D})}. \end{aligned} \tag{3.6}$$

Combinando tale decomposizione con l'*indipendenza dei parametri* si può riformulare la likelihood marginale (equazione 3.5) nel seguente modo:

$$\begin{aligned} P(\mathcal{D} \mid \mathcal{G}) &= \int_{\mathbf{q}_\mathcal{G}, \boldsymbol{\theta}_\mathcal{G}} L(\mathbf{q}_\mathcal{G} : \mathcal{D}) L(\boldsymbol{\theta}_\mathcal{G} : \mathcal{D}) P(\mathbf{q}_\mathcal{G}) P(\boldsymbol{\theta}_\mathcal{G})\, d\mathbf{q}_\mathcal{G} d\boldsymbol{\theta}_\mathcal{G} \\ &= \underbrace{\left[\int_{\mathbf{q}_\mathcal{G}} L(\mathbf{q}_\mathcal{G} : \mathcal{D}) P(\mathbf{q}_\mathcal{G})\, d\mathbf{q}_\mathcal{G}\right]}_{(a)} \cdot \underbrace{\left[\int_{\boldsymbol{\theta}_\mathcal{G}} L(\boldsymbol{\theta}_\mathcal{G} : \mathcal{D}) P(\boldsymbol{\theta}_\mathcal{G})\, d\boldsymbol{\theta}_\mathcal{G}\right]}_{(b)}. \end{aligned} \tag{3.7}$$

Ottenuta tale equazione, si affronta di seguito l'analisi e la decomposizione dei due termini che la compongono.

Utilizzando l'assunzione di *indipendenza locale dei parametri*, il termine (a) dell'equazione 3.7 è decomponibile nel seguente modo. Si noti che per brevità si pone $u = pa_i(x)$.

$$\prod_{X_i} \prod_{u} \prod_{x} \int_0^\infty P(\mathbf{q}_{x \mid u}) \cdot L_{X_i}(\mathbf{q}_{x \mid u} : \mathcal{D})\, d\mathbf{q}_{x \mid u}. \tag{a}$$

Sostituendo a tale termine la distribuzione a priori coniugata su $\mathbf{q}$ (si veda l'equazione 1.16) e la likelihood delle quantità di tempo trascorse in ogni stato (si veda l'equazione 1.10) si ottiene:

$$\prod_{X_i} \prod_{u} \prod_{x} \int_0^\infty \frac{(\tau_{x|u})^{\alpha_{x|u}+1}}{\Gamma(\alpha_{x|u}+1)} (\mathbf{q}_{x|u})^{\alpha_{x|u}} e^{-\mathbf{q}_{x|u}\tau_{x|u}} \cdot (\mathbf{q}_{x|u})^{M[x|u]} e^{-\mathbf{q}_{x|u}T[x|u]} d\mathbf{q}_{x|u}. \tag{a}$$

Si procede semplificando:

$$\prod_{X_i} \prod_{u} \prod_{x} \int_0^\infty \frac{(\tau_{x \mid u})^{\alpha_{x \mid u}+1} \cdot (\mathbf{q}_{x \mid u})^{\alpha_{x \mid u}+M[x \mid u]}}{\Gamma(\alpha_{x \mid u}+1) \cdot e^{\mathbf{q}_{x \mid u}(\tau_{x \mid u}+T[x \mid u])}}\, d\mathbf{q}_{x \mid u}. \tag{a}$$

E infine, risolvendo l'integrale, si ottiene:

$$\prod_{X_i} \underbrace{\prod_{u} \prod_{x} \frac{\Gamma(\alpha_{x|u} + M[x|u] + 1)(\tau_{x|u})^{\alpha_{x|u}+1}}{\Gamma(\alpha_{x|u} + 1)(\tau_{x|u} + T[x|u])^{\alpha_{x|u}+M[x|u]+1}}}_{MargL^{q}(X_i, Pa_{\mathcal{G}}(X_i) : \mathcal{D})} . \tag{a}$$

Relativamente all'analisi del termine (b) dell'equazione 3.7 si osservi che, poiché le distribuzioni sui parametri θ sono di *Dirichlet*, tale operazione è analoga a quella comune per le Bayesian Network.

Ne consegue che il termine (b) si semplifica:

$$\prod_{X_i} \underbrace{\prod_{u} \prod_{x} \frac{\Gamma(\alpha_{x|u})}{\Gamma(\alpha_{x|u} + M[x|u])} \cdot \prod_{x \neq x'} \frac{\Gamma(\alpha_{xx'|u} + M[x, x'|u])}{\Gamma(\alpha_{xx'|u})}}_{MargL^{\theta}(X_i, Pa_{\mathcal{G}}(X_i) : \mathcal{D})} . \tag{b}$$

Quindi si può riformulare la likelihood marginale:

$$P(\mathcal{D}|\mathcal{G}) = \prod_{X_i} MargL^{q}(X_i, Pa_{\mathcal{G}}(X_i) : \mathcal{D}) \cdot MargL^{\theta}(X_i, Pa_{\mathcal{G}}(X_i) : \mathcal{D}). \tag{3.8}$$

Al fine di derivare la probabilità a priori della struttura (equazione 3.2) si è già assunto in precedenza che l'ipotesi di modularità della struttura sussista. Perciò, sfruttando tale assunzione, l'equazione 3.2 della probabilità a priori della struttura, e l'equazione 3.8 della likelihood marginale:

$$\begin{aligned} score_B(\mathcal{G} : \mathcal{D}) = \sum_{X_i} \Big[& \ln P(Pa(X_i) = Pa_{\mathcal{G}}(X_i)) + \\ & + \ln MargL^{q}(X_i, Pa_{\mathcal{G}}(X_i) : \mathcal{D}) + \\ & + \ln MargL^{\theta}(X_i, Pa_{\mathcal{G}}(X_i) : \mathcal{D}) \Big] = \\ = \sum_{X_i} & famscore_{\mathcal{B}}(X_i, Pa_{\mathcal{G}}(X_i) : \mathcal{D}). \end{aligned} \tag{3.9}$$

Si è quindi definita la funzione di scoring come una somma di score bayesiani, $famscore_{\mathcal{B}}(X_i, Pa_{\mathcal{G}}(X_i) : \mathcal{D})$, relativi ai nodi del grafo $\mathcal{G}$. Ognuno di tali score bayesiani misura la qualità di $Pa_{\mathcal{G}}(X_i)$ come insieme dei nodi genitori di X_i, dato l'insieme dei dati di apprendimento $\mathcal{D}$.

3.2 RICERCA DELLA STRUTTURA

In questa sezione si affronta il secondo passo del processo di apprendimento strutturale: l'utilizzo di una *procedura di ottimizzazione* finalizzata alla ricerca di una struttura che massimizzi lo *score bayesiano*.

Chickering (1994) ha mostrato come il problema di apprendere la struttura ottimale di una Bayesian Network, detto problema *k-learn*, dove k è il numero massimo di genitori per ogni variabile casuale, sia un problema *NP-arduo* anche qualora si imponga $k = 2$. La ragione di tale complessità è dovuta al vincolo di aciclicità delle BN (i. e., il grafo di una BN, come da definizione 1.1, deve essere un DAG): non è perciò possibile determinare l'insieme ottimale dei genitori di ogni nodo di una BN individualmente; poiché la scelta di un insieme di genitori per un nodo restringe la possibilità di scelta relativa ai nodi restanti.

Come già accennato ed intuibile dalla composizione dello score bayesiano (si veda la funzione $famscore_{\mathcal{B}}$, equazione 3.9), la ricerca della struttura ottimale di un modello Continuos time Bayesian Network (CTBN) è invece notevolmente più semplice rispetto a quello relativo alle BN (o alle DBN). La motivazione di tale vantaggio risiede nel fatto che, poiché gli archi fra i nodi del grado $\mathcal{G}$ di una CTBN rappresentano l'effetto del valore attuale della variabile casuale padre sul valore successivo della variabile casuale figlia, non esiste un vincolo di aciclicità ed è di conseguenza possibile ottimizzare l'insieme dei nodi genitori di un qualsiasi nodo separatamente dagli altri.

Inoltre, qualora si restringa il massimo numero di genitori a un valore k, per ogni variabile casuale X_i di una CTBN, con $i = 1, \ldots, N$, si può semplicemente enumerare ogni suo possibile insieme di nodi genitori $Pa(X_i)$ tale che $|Pa(X_i)| \leqslant k$ e calcolarne il rispettivo punteggio $famscore_{\mathcal{B}}(X_i, Pa(X_i) : \mathcal{D})$. Infine scegliere come insieme dei nodi genitori di X_i quello con punteggio massimo.

Si definisce perciò il seguente teorema (Nodelman *et al.*, 2002).

Teorema 3.1 (Problema k-learn)**.** *Il problema* ***k-learn*** *per le Continuos time Bayesian Network, fissato* k*, può essere risolto in tempo polinomiale rispetto al numero di variabili casuali* N *e alla dimensione dell'insieme di dati* $\mathcal{D}$*.*

Si osservi che, fissando k a priori, non è necessario enumerare esaustivamente tutti i possibili insiemi di nodi genitori di ogni nodo di una CTBN. É quindi possibile utilizzare un algoritmo di ricerca euristica di tipo *greedy* per esplorare lo spazio di ricerca. Nella sottosezione 3.2.1 si presenta l'algoritmo scelto per l'apprendimento strutturale della struttura ottimale di un modello CTBN.

3.2.1 Hill Climbing

L'algoritmo *hill climbing* è una tecnica di ricerca euristica finalizzata alla risoluzione di problemi di ottimizzazione i cui stati (i. e., elementi dello spazio di ricerca) contengono tutte le informazioni necessarie a costituire una soluzione (Russell e Norvig, 2003).

L'idea su cui si basa tale algoritmo consiste nell'iniziare la ricerca con una soluzione sub-ottimale e, nei passi successivi, migliorare ite-

rativamente la soluzione finché una qualche condizione di fermata venga raggiunta (e. g., l'algoritmo non può migliorare ulteriormente la soluzione corrente). Il processo di miglioramento è, come già accennato, basato sulla valutazione dello stato corrente tramite una funzione di punteggio.

A differenza di altri algoritmi di ricerca basati sul miglioramento iterativo della soluzione (e. g., *simulated annealing*, *tabu search*), l'algoritmo *hill climbing* si sposta sempre in uno stato che fornisce una soluzione con maggior punteggio (Russell e Norvig, 2003). L'utilizzo di tale algoritmo garantisce il raggiungimento di soluzioni localmente ottime (i. e., soluzioni che non possono essere migliorate considerando solamente la configurazione degli stati vicini) in una quantità di tempo relativamente bassa. Tuttavia esso non garantisce il raggiungimento di una soluzione che costituisca l'ottimo globale, a meno che la funzione rappresentante lo spazio di ricerca non sia convessa. Ciò avviene poiché l'algoritmo smette di effettuare progressi verso la soluzione globalmente ottima nel momento in cui il vicinato della soluzione corrente non permette alcun miglioramento immediato.

Per sorpassare tale limitazione è possibile attuare varie strategie (si veda Russell e Norvig, 2003), quali, ad esempio: l'esplorazione iterativa, a partire da diverse configurazioni iniziali, dello stesso spazio di ricerca (i. e., *random restart hill climbing*); la selezione stocastica del vicinato da esaminare ad ogni passo della ricerca locale, sulla base della probabilità che un dato vicinato porti a un progresso maggiore rispetto ad altri vicinati; oppure la scelta di soluzioni non migliorative finalizzata ad una maggiore esplorazione dello spazio di ricerca (i. e., *simulated annealing*).

Nota 3.2.1.1. Questo algoritmo opera una gestione efficiente della memoria poiché non necessita il mantenimento di alcun albero di ricerca: esso conserva solamente l'informazione (i. e., punteggio della soluzione) sullo stato corrente e quello successivo.

L'algoritmo 3.1 illustra la tecnica di ricerca *hill climbing* dato uno spazio degli stati discreto.

```
1  function hillclimbing(problem) {
2      var current_state = start_state(problem)
3      while (true) {
4          var nb = neighbors(current_state)
5          var next_eval = -inf
6          var next_state = null
7          for (x in nb) {
8              var x_score = score(x)
9              if (x_score > next_eval) {
10                 next_state = x
11                 next_eval = x_score
12             }
```

```
13          }
14          if (next_eval ⩽ score(current_state)) {
15              break
16          }
17          current_state = next_state
18      }
19      return current_state
20  }
```

Algoritmo 3.1: Algoritmo *hill climbing* per uno spazio degli stati discreto.

Di seguito si discute l'applicazione di tale algoritmo all'apprendimento strutturale delle CTBN.

Come detto, a causa della mancanza del vincolo di aciclicità, è possibile eseguire la succitata procedura di ottimizzazione in modo indipendente per ogni singolo nodo del modello CTBN in esame. Ciò permette di scomporre il problema dell'apprendimento strutturale in un insieme di problemi della stessa entità, di minore complessità e completamente indipendenti fra di essi; contesto ottimo per un approccio parallelizzato alla risoluzione del problema padre.

Dato uno qualsiasi dei succitati sotto-problemi, lo spazio degli stati che l'algoritmo *hill climbing* deve esplorare è composto da tutti i possibili insiemi di genitori con cardinalità minore o uguale a k, il numero massimo di genitori di ogni singolo nodo della CTBN. L'algoritmo *hill climbing* individua l'insieme di genitori ottimale per il nodo in esame, valutando, iterativamente, i possibili insiemi di nodi genitori con cardinalità minore e maggiore di 1 rispetto alla cardinalità dell'insieme di genitori corrente. Si osservi che tale configurazione prevede che l'insieme dei genitori con cardinalità pari a 0 (i. e., insieme vuoto $\emptyset$) venga anch'esso valutato.

La funzione di *scoring* con cui tali insiemi di genitori vengono valutati, rappresentata nell'algoritmo 3.1 dalla funzione **score**, relativamente all'apprendimento strutturale di una CTBN, corrisponde alla funzione $famscore_{\mathcal{B}}$, presentata nella sezione 3.1 a pagina 34.

4 PACKAGE R

...

4.1 ANALISI

...

4.2 PACKAGE CTBN

...

5 STRUMENTI PER LA CREAZIONE DI DATASET

Al fine di valutare le prestazioni degli algoritmi di apprendimento e classificazione delle Continuos time Bayesian Network è emersa la necessità di generare dei dataset adeguati a tale scopo.

Come già specificato in precedenza, le CTBN sono un modello stocastico dedito alla rappresentazione dell'evoluzione di sistemi dinamici, cioè di fenomeni cangianti nel tempo (non necessariamente o esclusivamente in base ad esso) rappresentabili come insiemi di traiettorie multi-variate. Si è quindi scelto di generare dei dataset che rappresentassero un tipico sistema cangiante: il traffico automobilistico su rete stradale.

Tali dataset, costituiti da un insieme di documenti rappresentanti la presenza (durante l'evolvere del tempo) di veicoli sui sensori di una rete stradale, sono stati generati con l'ausilio di un software commerciale, Traffic Software Integrated System (TSIS) (versione $\geqslant$ 6.2) e di una sua estensione a tempo d'esecuzione appositamente sviluppata al fine di monitorare e tracciare il passaggio dei veicoli.

In questo capitolo si presentano sia i succitati strumenti utilizzati per la creazione di reti stradali e relativi modelli di simulazione, sia lo strumento creato per la creazione di dataset relativi al traffico.

Relativamente, invece, al processo pratico di creazione dei dataset si rimanda alla sezione A.2.

5.1 TSIS

Traffic Software Integrated System (TSIS) è un ambiente di sviluppo integrato[16], distribuito commercialmente da McTrans[17] e supportato dalla Federal Highway Administration (FWHA)[18], il cui scopo ultimo è permettere la simulazione e l'analisi di modelli di reti stradali.

TSIS è costituito da insieme di strumenti dedicati alla creazione di reti stradali e relativi modelli di simulazione, all'esecuzione, e eventualmente alla visualizzazione, di tali modelli, così come all'in-

16 Un ambiente di sviluppo integrato, comunemente chiamato anche Integrated Development Environment (IDE), è un insieme di programmi finalizzati a supportare il processo di sviluppo dei software. Generalmente, un IDE è costituito da uno strumento per la creazione e modifica del codice sorgente, un compilatore o un interprete, strumenti per l'automazione dello sviluppo e la qualità del codice sorgente.

17 McTrans Moving Technology: `http://mctrans.ce.ufl.edu`.

18 Agenzia del Dipartimento dei Trasporti degli Stati Uniti d'America: `www.fhwa.dot.gov`.

terpretazione dei risultati ottenuti. Tale insieme di strumenti è reso accessibile tramite delle interfaccie grafiche[19].

L'architettura modulare con cui TSIS è realizzato permette, in caso di necessità, di estendere tale ambiente creando degli ulteriori strumenti.

Di seguito si introducono brevemente i concetti relativi a TSIS utilizzati nel prosieguo di questo lavoro di tesi.

Tuttavia si osservi che, poiché lo scopo di questa sezione non consiste nel documentare TSIS, la sua trattazione esaustiva (e. g., semantica e lista completa dei tipi di dati rappresentabili) è omessa. A tale fine si rimanda invece alla documentazione ufficiale del software in questione.

Definizione 5.1 (Progetto TSIS)**.** Un progetto TSIS è un insieme di modelli di simulazione per una specifica rete stradale.

Definizione 5.2 (Modello di simulazione TSIS)**.** Un modello di simulazione è costituito da un input (e. g., variazioni dei flussi di ingresso nella rete, variazioni delle percentuali di svolta dei veicoli nelle intersezioni) per la simulazione di una determinata rete stradale e tutti i dati generati dalla sua esecuzione (i. e., simulazione).

Osservazione 5.2.1. Un modello di simulazione può anche prevedere che la sua esecuzione sia eseguita più volte. Finché il seme dei numeri casuali non è modificato, esso è sempre considerato un singolo modello di simulazione.

Definizione 5.3 (Formato TRF)**.** TRF[20] è il formato dei file accettati dal simulatore di TSIS. Esso codifica e rappresenta una rete stradale e il relativo modello di simulazione specificandone i vari componenti tramite l'utilizzo dei rispettivi RT (si veda la definizione 5.5).

Osservazione 5.3.1. Il formato TRF è equivalente al formato TRAF[21].

Definizione 5.4 (Formato TNO)**.** TNO[22] è il formato nativo con cui vengono rappresentate in memoria le reti stradali create visivamente tramite l'interfaccia grafica TRAFED.

Definizione 5.5 (Record Type)**.** Un Record Type (RT) rappresenta il blocco informativo minimo su cui è costruita una rete stradale e il suo modello di simulazione. Nel concreto esso consiste in una singola riga di testo nei file TRF contenente un codice identificativo numerico e dei valori per i rispettivi campi accettati nell'ordine prestabilito. Allo stesso modo, tale formato descrive anche il modello di simulazione della rete stradale.

19 Un'interfaccia grafica, nota anche come Graphical User Interface (GUI), è un tipo di interfaccia utente il cui fine è permettere all'utente di interagire con il software manipolando oggetti grafici convenzionali.

20 Traffic File (TRF).

21 Traffic File (TRAF).

22 TRAFED Native Object (TNO).

Definizione 5.6 (RTE)**.** Una Run-Time Extension (RTE), estensione a tempo d'esecuzione, è un'applicazione in grado di comunicare a tempo d'esecuzione con un'altra applicazione esterna. Una RTE, in ambiente *Microsoft Windows*, è solitamente compilata separatamente sotto forma di DLL[23] e deve rispettare una determinata interfaccia: deve perciò implementare e esportare determinate funzioni che l'applicazione oggetto della comunicazione chiama a tempo d'esecuzione. Per la comunicazione in ingresso, invece, una RTE deve essere collegata alle librerie dell'applicazione con cui intende interfacciarsi, avendo così accesso alle strutture dati e alle funzioni che questa esporta.

5.1.1 Componenti

In questa sezione si elencano gli strumenti che costituiscono l'ambiente di sviluppo TSIS.

CORSIM

CORSIM[24] costituisce il componente principale dell'insieme di strumenti denominato TSIS. È un simulatore il cui obiettivo è permettere la creazione e l'esecuzione di modelli di simulazione TSIS. È composto da due simulatori integrati che rappresentano l'intero sistema di traffico come funzione del tempo: NETSIM[25] e FRESIM[26]. Tali simulatori integrati rappresentano, rispettivamente, il traffico sulle strade urbane e non. La simulazione effettuata da tali strumenti è di tipo microscopico: essi modellano individualmente il comportamento di ogni singolo veicolo, prendendo in considerazione per ognuno di essi una serie di variabili, anche di tipo stocastico (e. g., tipologia di guidatore). Per tale motivo CORSIM è dotato di molte possibili opzioni di configurazione e permette lo studio di modelli molto complessi e dettagliati.

TRAFED

TRAFED[27] è una GUI il cui scopo è permettere la creazione e la modifica di reti stradali e di modelli di simulazione per CORSIM.

TSHELL

TShell[28] è la GUI di TSIS. Funge da contenitore degli strumen-

23 Una libreria a collegamento dinamico (DLL) è una libreria software che viene caricata dinamicamente in fase di esecuzione, invece di essere collegata staticamente a un eseguibile in fase di compilazione. Queste librerie sono anche chiamate Dynamic-link library (DLL). L'acronimo DLL corrisponde all'estensione che tali oggetti hanno in ambiente *Microsoft Windows*. Tuttavia esse sono spesso chiamate più genericamente con il termine librerie condivise (da *shared library*). Nei sistemi *Linux*, esse sono anche note come oggetti *shared object*.

24 Corridor microscopic simulation program (CORSIM).

25 Network Simulator (NETSIM).

26 Freeway Simulator (FRESIM).

27 TRAF Editor (TRAFED).

28 TSIS Shell (TShell).

ti (preconfigurati, o creati dall'utente) di questo ambiente di sviluppo integrato e permette la gestione dei progetti TSIS.

TRAFVU

TRAFVU[29] è una GUI finalizzata alla visualizzazione dei modelli di simulazione simulati con CORSIM. Essa permette sia di visualizzare in modo animato l'evoluzione del traffico nella rete stradale con una qualsiasi granularità temporale, sia di visualizzare una serie di misure di interesse relative alla simulazione.

TSIS TEXT EDITOR

TSIS Text Editor è uno strumento il cui scopo è facilitare la modifica manuale dei file TRF. A tale scopo esso visualizza per ogni RT che si intende modificare sia la sua descrizione sia l'insieme dei campi supportati.

TSIS SCRIPT TOOL

TSIS Script Tool è uno strumento per la creazione, la modifica e l'esecuzione di codice VBScript[30]. Questo strumento fornisce un meccanismo utile ad automatizzare le funzionalità di simulazione di TSIS (e. g., esecuzioni multiple dello stesso modello di simulazione variando il seme dei numeri casuali che governa la distribuzione di ingresso dei veicoli nella rete stradale).

TSIS TRANSLATOR

TSIS Translator è uno strumento utile alla conversione dei file dal formato TRF al formato TNO e viceversa. Tale operazione risulta utile al fine di rendere i file TRF utilizzabili tramite lo strumento TRAFED così come per rendere i file TNO utilizzabili con CORSIM.

TSIS OUTPUT PROCESSOR

TSIS Output Processor è uno strumento finalizzato alla raccolta e l'aggregazione dei dati da CORSIM durante l'esecuzione multipla di modelli di simulazione. La sua caratteristica principale consiste nella computazione automatica di un insieme di statistiche predefinite. Esso permette di scegliere sia le statistiche di interesse sia la granularità temporale della loro computazione.

5.1.2 Caratteristiche

Segue una panoramica il cui scopo è presentare sia le principali capacità, sia i vincoli di modellazione, simulazione e analisi di TSIS.

TSIS, tramite CORSIM, permette la modellazione di reti stradali con le seguenti caratteristiche:

29 TRAF Visualization Utility (TRAFVU).

30 Microsoft's Visual Basic Scripting Edition (VBScript) è un linguaggio interpretato, sottoinsieme del linguaggio *Visual Basic*, utilizzato come sostituto o integrazione della linea di comando o per il controllo di applicazioni esterne in ambiente *Microsoft Windows*.

- reti stradali urbane, non urbane o miste
- controllo quantitativo (i. e., quantità di veicoli per intervallo temporale, distribuita secondo una distribuzione statistica) e qualitativo (i. e., tipo di veicoli) dei flussi di veicoli in ingresso nella rete stradale
- controllo completo del flusso di traffico (i. e., percentuali di svolta dei veicoli variabile nel tempo)
- supporto per strade con più corsie (massimo 9)
- supporto dei segnali stradali
- supporto per gli attraversamenti pedonali
- intersezioni non controllate
- intersezioni controllate da semafori
- canalizzazione delle corsie
- piani semaforici predefiniti, dinamici (e. g., priorità ai veicoli di emergenza) o guidati da algoritmi (e. g., attivazione del semaforo in base alle rilevazioni effettuate dai sensori)
- sensori per la rilevazione del passaggio o della presenza di veicoli
- supporto per i mezzi di trasporto pubblico (i. e., autobus, taxi)
- rilevamento di incidenti
- supporto per i sistemi di guida anglosassoni
- simulazione guidata all'analisi delle code
- simulazione guidata allo studio del grado di occupazione della rete stradale
- output di dati di interesse aggregati
- output di dati statistici collezionati

Poiché un modello di simulazione del traffico è caratterizzato dal cambiamento di un insieme di condizioni (e. g., volumi di traffico, canalizzazione delle corsie, percentuali di svolta dei veicoli) della rete stradale, esso deve specificare, oltre alla natura stessa dei cambiamenti, anche i fattori in base a cui essi avvengono. Il fattore primario che viene preso in considerazione è il tempo. Ne consegue perciò che un modello di simulazione del traffico deve specificare l'intervallo temporale in cui specifici cambiamenti di determinate condizioni avvengono.

CORSIM affronta questo problema permettendo di partizionare il tempo totale di simulazione in una serie di periodi temporali (i. e., *time period*) di durata variabile. Ogni periodo temporale possiede perciò un insieme di dati di input che non variano per tutta la sua durata. Inoltre, i periodi temporali sono a loro volta suddivisi in intervalli temporali (i. e., *time interval*), anch'essi suddivisi in passi temporali (i. e., *time step*).

Le principali limitazioni di TSIS derivano dalle modalità con cui CORSIM implementa la gestione del tempo: la lista dei periodi temporali è rappresentata internamente tramite un array statico di dimensione prefissata. Nello specifico, un modello CORSIM può essere costituito da un massimo di 19 periodi temporali, ognuno dei quali viene specificato tramite il RT 03, e può avere una durata compresa tra i 10 e i 9999 secondi. Ognuno di tali periodi temporali è suddiviso in intervalli (specificati tramite dei RT 04) la cui durata, compresa tra 1 e 200 secondi, deve essere un sottomultiplo della durata del periodo temporale cui appartiene. Infine, ogni intervallo temporale è partizionato in passi temporali: NETSIM utilizza un passo temporale fisso di 1 secondo mentre FRESIM opera in base al passo temporale specificato dall'utente. Poiché i due modelli di simulazione microscopica che costituiscono CORSIM operano in modo sincronizzato, la durata del passo temporale di FRESIM, anche se compresa nell'intervallo $[0.1, 1]$ secondi, non può essere specificata liberamente dall'utente. All'utente è permesso specificare solo il numero di passi temporali di FRESIM (tramite il campo 1, RT 04) che devono essere eseguiti per ogni secondo di simulazione. Tale approccio permette perciò di utilizzare dei passi temporali di durata minore per il simulatore FRESIM mantenendo la sincronizzazione con il simulatore NETSIM. La tabella 5.1 mostra la corrispondenza tra il numero di passi temporali di FRESIM e la reale durata che verrà loro assegnata.

Valore del campo 1	**Durata del passo temporale in FRESIM** (sec)
1	1.0
2	0.5
3	0.333333
4	0.25
5	0.2
6	0.166667
7	0.142857
8	0.125
9	0.111111
10	0.1

Tabella 5.1: Relazione tra numero di passi temporali per ogni secondo di simulazione e durata effettiva del passo temporale in FRESIM.

La figura 5.1 mostra, tramite un esempio, quanto appena descritto relativamente alla rappresentazione del tempo in CORSIM.

Si osservi, inoltre, che la granularità temporale massima con cui è possibile ottenere statistiche cumulative, e, in generale, dati relativi alla simulazione, corrisponde alla durata scelta per gli intervalli temporali (si veda a tal riguarda la documentazione relativa al tipo RT 05). Si consideri ad esempio la figura 5.1: in tal caso non sarà possibile ottenere dati cumulati relativi a un intervallo di tempo minore di 60 secondi.

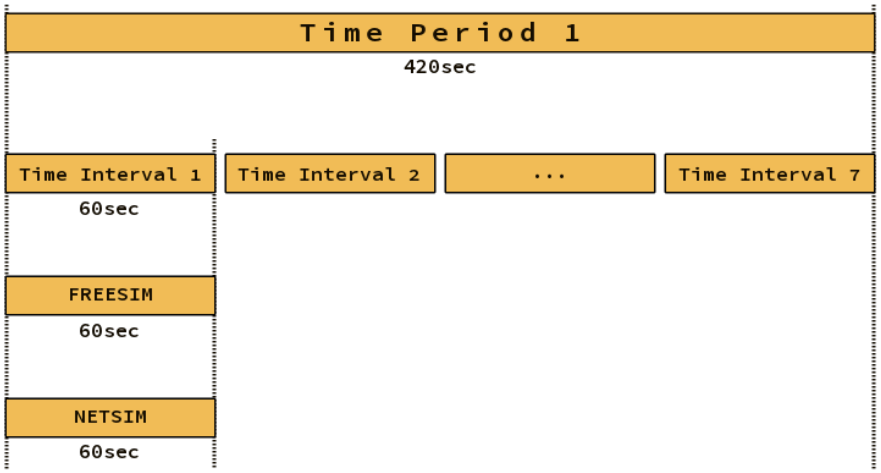


Figura 5.1: Esempio di suddivisione gerarchica delle unità temporali in CORSIM. Un *time period* della durata di 420 secondi è suddiviso in 7 *time interval*, ciascuno della durata di 60 secondi. Ogni *time interval* verrà poi automaticamente diviso in 60 *time step* da 1 secondo ciascuno per NETSIM. La durata di tali *time step* di NETSIM può o meno coincidere con quella assegnata ai *time step* di FRESIM a seconda del valore assegnato al campo 1 del RT 04.

La gestione del tempo che CORSIM attua implica quindi alcune limitazioni:

- non è possibile eseguire modelli di simulazione che si protraggono oltre un tempo di circa 52 ore
- la massima granularità con cui è possibile raccogliere informazioni è di 1 secondo, impostando a tale valore la durata degli intervalli temporali
- i dati raccolti dalla simulazione sono sempre aggregati in base alla durata dell'intervallo temporale cui si riferiscono.

5.2 CREAZIONE DI ESTENSIONI TSIS

Come accennato, TSIS espone un meccanismo finalizzato all'estensione delle sue funzionalità tramite la creazione, da parte dell'uten-

te, di altri strumenti da integrare nell'ambiente di sviluppo. Tali strumenti, interfacciandosi direttamente con CORSIM, possono modificarne o aumentarne la logica di simulazione, collezionare dati o monitorare eventi speciali (e. g., indicenti).

In questa sottosezione si presenta il funzionamento dei meccanismi di interfacciamento (i. e., più brevemente detti API[31]) tra CORSIM e strumenti esterni, a cui ci si riferirà da questo momento in poi con il termine CORSIM RTE.

5.2.1 Requisiti

Al fine di sviluppare e compilare con successo una CORSIM RTE in C++ è necessario disporre dei seguenti strumenti:

- il compilatore della piattaforma *Microsoft Visual C++*
- il pacchetto software di TSIS, il quale include di default tutti i componenti necessari mostrati dalla figura 5.2 nella pagina successiva (ad eccezione, chiaramente, del componente RTE).

Si osservi, inoltre, che è possibile sviluppare una CORSIM RTE anche in linguaggio `C` o `FORTRAN`.

5.2.2 Architettura di CORSIM

La figura 5.2 nella pagina seguente illustra l'architettura modulare di CORSIM e il suo funzionamento all'interno dell'ambiente di sviluppo TSIS. Si osservi che i componenti che costituiscono CORSIM sono di due tipi: librerie DLL e moduli COM[32]. Il CORSIM Driver Component, ad esempio, è il modulo COM di TSIS preposto a interfacciare CORSIM e TShell, permettendo così il controllo e l'esecuzione di CORSIM, delle RTE create dall'utente e degli altri strumenti (e. g., TSIS Output Processor) di TSIS tramite GUI.

Anche se la figura 5.2 nella pagina successiva mostra per completezza l'intera architettura di CORSIM, si procede con la descrizione delle interfacce di CORSIM preposte alla comunicazione con RTE sviluppate dall'utente, identificate dalle frecce tratteggiate e numerate.

Per ogni passo temporale di simulazione, il CORSIM Server chiama una serie di funzioni di CORSIM finalizzate a guidare l'andamento della simulazione. Quando una RTE viene inserita nell'ambiente di sviluppo integrato, il CORSIM Server chiama anche le funzioni che la RTE

31 Con il termine Application Programming Interface (API) si indica un insieme di procedure rese disponibili all'esterno, di solito raggruppate a formare un insieme di strumenti specifici per l'espletamento di un determinato compito all'interno di un programma.

32 Il Component Object Model (COM) è uno standard per componenti software ideato da *Microsoft*. Il suo fine consiste nel permettere la comunicazione fra processi e la creazione dinamica di oggetti. Una interfaccia COM è una collezione di funzioni, incapsulata in un componente software binario e neutrale rispetto al linguaggio.

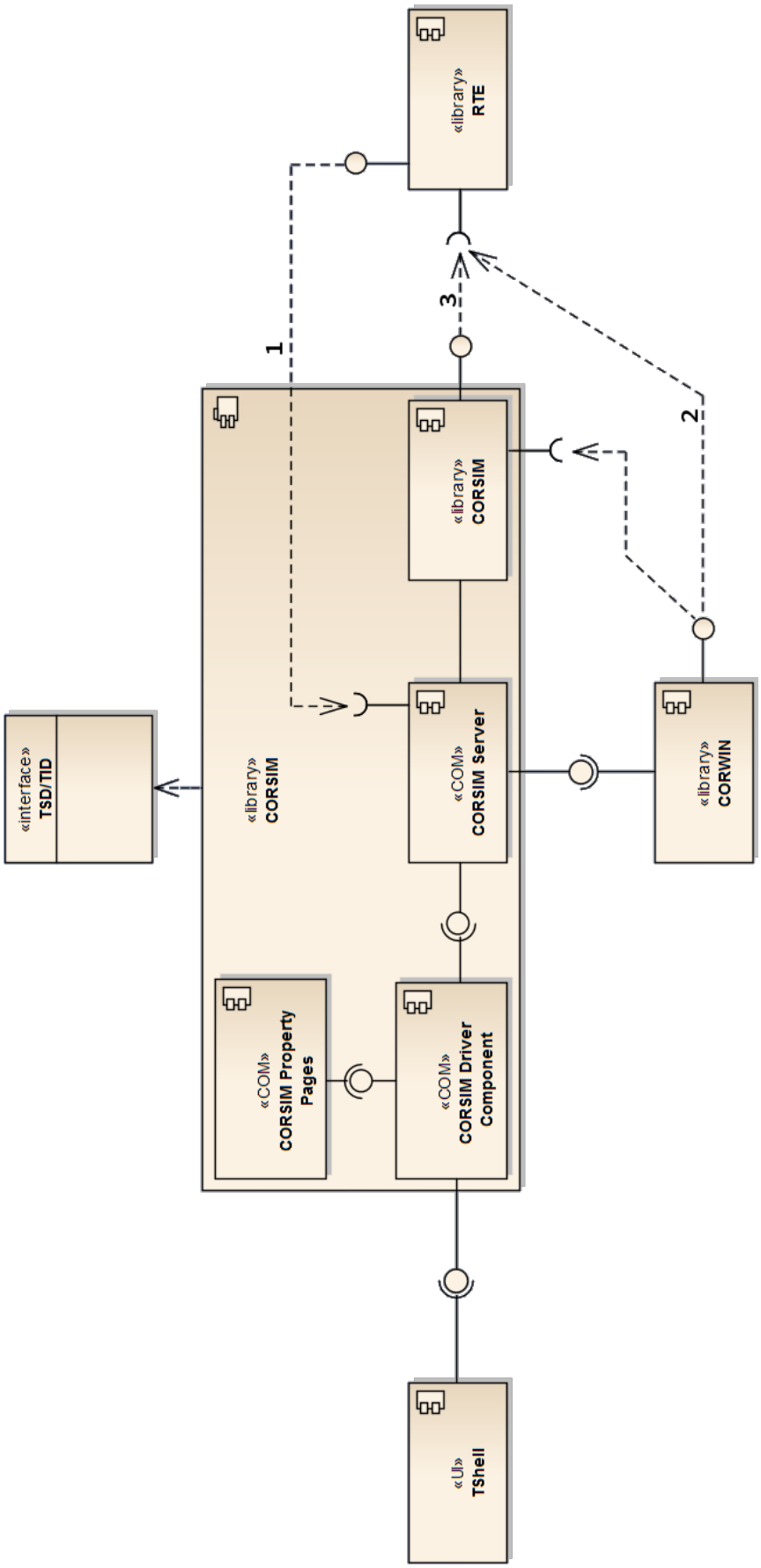


Figura 5.2: Porzione del diagramma dei componenti di TSIS: mostra l'architettura modulare e il funzionamento di CORSIM all'interno di TSIS.

esporta in base ai messaggi che riceve da CORSIM durante la sua esecuzione. Questa interfaccia è rappresentata dalla freccia 1 nella figura 5.2 nella pagina precedente; la sottosezione 5.2.3 in questa pagina riporta maggiori dettagli sui punti di chiamata che CORSIM espone.

TSIS fornisce inoltre una API, identificata dalla freccia 2 nella figura 5.2 nella pagina precedente, chiamata CORWIN, che permette alle RTE di inviare messaggi al modulo CORSIM Server affinché essi siano visualizzati in TShell dal CORSIM Driver Component.

Infine, una RTE può accedere direttamente a una serie di funzioni e strutture dati esportate da CORSIM nella memoria condivisa. Anche se non è possibile riferirsi a questo meccanismo di comunicazione come una API vera e propria, nel prosieguo, ci riferiremo ad essa con il termine CORSIM API al fine di semplificare la discussione. Perciò, la CORSIM API, rappresentata dalla freccia 3 della figura 5.2 nella pagina precedente, oltre a permettere l'estrazione di informazioni relative alla simulazione, permette alla RTE di controllare, eventualmente, molti aspetti della simulazione operata da CORSIM (e. g., aborto della simulazione).

Si osservi che, anche se la figura 5.2 nella pagina precedente non evidenzia tale possibilità, l'architettura di CORSIM supporta l'utilizzo di più RTE contemporaneamente.

Nota 5.2.1. Un attento osservatore noterà come CORSIM, la libreria finalizzata al processo di simulazione, sia a sua volta una RTE automaticamente collegata ai moduli CORSIM Server e CORWIN. Inoltre, la figura 5.3 a pagina 52 fa notare come tutti i componenti (elencati e descritti nella sottosezione 5.1.1 a pagina 43) dell'ambiente di sviluppo TSIS siano anch'essi dei moduli architetturalmente uguali alle RTE.

5.2.3 Ciclo di vita di CORSIM

Di seguito si presenta il ciclo di vita di CORSIM descrivendo i punti di chiamata che esso esporta tramite apposite funzioni affinché una RTE, implementando ed esportando una funzione per almeno uno di essi, possa interfacciarsi con il processo di simulazione. Tali informazioni sono quindi relative alla API rappresentata dalla freccia 1 nella figura 5.2 nella pagina precedente. Le modalità di implementazione e utilizzo in C++ di tale API sono illustrate nella sottosezione 5.2.5 a pagina 55.

La tabella 5.2 nella pagina successiva descrive tutti i punti di chiamata relativi alla linea di esecuzione temporale di CORSIM.

Punto di chiamata	Descrizione
`Initialize`	Chiamato all'inizio della simulazione prima della fase di inizializzazione CORSIM ma dopo la lettura del file TRF di input.
`PostVehicleEmit`	Chiamato ad ogni *time step* dopo che i veicoli sono stati immessi nella rete stradale.
`PreNetsimVehicle`	Chiamato ad ogni *time step* appena prima che i veicoli inizino a muoversi nel sotto-modello NETSIM.
`PreNetsimSignal`	Chiamato ad ogni *time step* appena prima che i segnali (e. g., semafori) del sotto-modello NETSIM vengano impostati.
`PostNetsimTimestep`	Chiamato in corrispondenza della fine del processo di simulazione di ogni *time step* di NETSIM e prima che FRESIM inizi a simulare il suo relativo sotto-modello.
`PreFresimVehicle`	Chiamato ad ogni *time step* appena prima che i veicoli inizino a muoversi nel sotto-modello FRESIM.
`PreFresimSignal`	Chiamato ad ogni *time step* appena prima che i segnali (e. g., semafori) del sotto-modello FRESIM vengano impostati.
`PostFresimTimestep`	Chiamato in corrispondenza della fine del processo di simulazione di ogni *time step* di FRESIM.
`BeginSimulation`	Chiamato dopo l'inizializzazione di CORSIM (i. e., rete stradale piena) in corrispondenza dell'inizio della simulazione.
`TimeStepComplete`	Chiamato in corrispondenza della fine del processo di simulazione di ogni *time step*.
`TimeIntervalComplete`	Chiamato in corrispondenza della fine del processo di simulazione di ogni *time interval*.
`TimePeriodComplete`	Chiamato in corrispondenza della fine del processo di simulazione di ogni *time period*.
`TimePeriodValidated`	Chiamato in corrispondenza della fine del processo di lettura e validazione del file di input relativo a ogni *time period*, prima dell'inizializzazione e dell'effettivo inizio del processo di simulazione di ogni *time period*.
`SimulationComplete`	Chiamato in corrispondenza della fine della simulazione e prima della completa terminazione del processo.
`Shutdown`	Chiamato appena prima che l'esecuzione di CORSIM termini.
`Exit`	Chiamato in corrispondenza della fine dell'intero processo di simulazione.

Tabella 5.2: Descrizione di punti di chiamata che CORSIM espone all'esterno.

Una volta compilata la RTE e ottenuto il relativo file DLL, ognuno

dei punti di chiamata di CORSIM deve essere associato alla rispettiva funzione implementata dalla RTE. La sottosezione 5.2.4 in questa pagina descrive in maggior dettaglio tale processo.

5.2.4 Collegare una RTE a CORSIM

Questa sottosezione illustra i passi necessari a espletare il processo di collegamento (i. e., *linking*) di una RTE a CORSIM. Tale processo è eseguibile direttamente tramite TShell.

A scopo esemplificativo si descrive come aggiungere la RTE per il rilevamento e tracciamento del passaggio dei veicoli sui sensori (descritta nella sezione 5.3 a pagina 57). Tale operazione viene svolta tramite il menù *Tools* di TShell scegliendo la voce *Tool Configuration* e cliccando sul pulsante per l'aggiunta (i. e., *Add*) di una RTE. La figura 5.3 mostra lo strumento di configurazione degli strumenti appartenenti all'ambiente di TSIS.

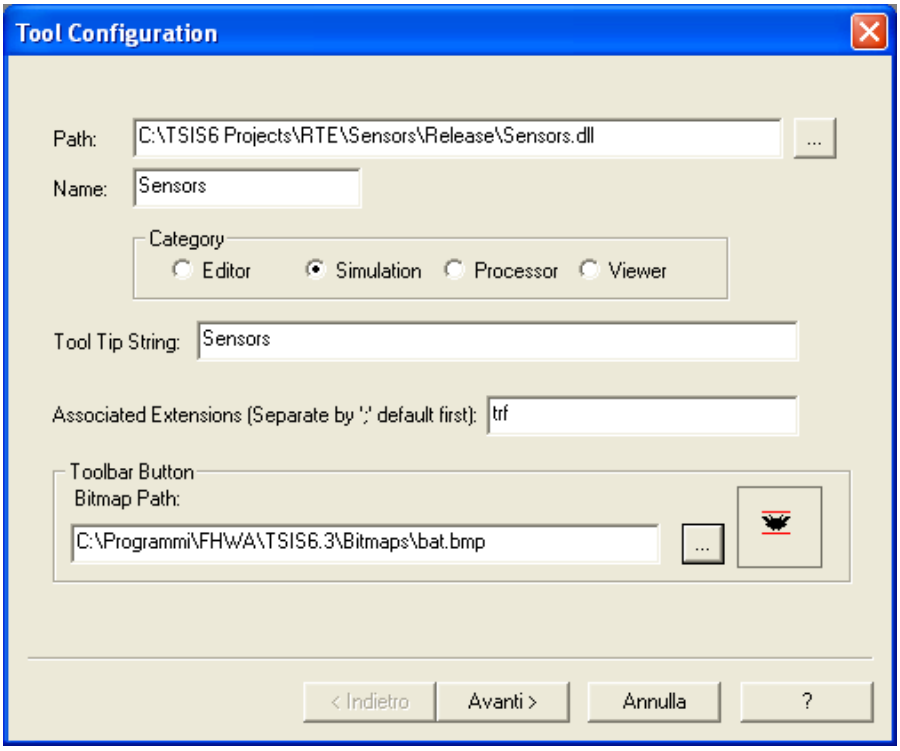


Figura 5.3: Strumento finalizzato all'aggiunta di una RTE a TSIS.

Specificato il percorso a cui risiede il file DLL della RTE, il tipo di RTE, il nome e l'icona che si intende assegnare a tale strumento e il tipo di file a cui va associato (e. g., TRF), la RTE è aggiunta a TSIS. La figura 5.4 nella pagina successiva mostra la barra degli strumenti di TShell quando un file TRF viene aperto: essa contiene il pulsante per l'avvio della RTE appena aggiunta all'ambiente di sviluppo TSIS.

Figura 5.4: Barra degli strumenti di TShell contenente il pulsante per l'invocazione della RTE aggiunta all'ambiente di sviluppo TSIS.

Tuttavia, come detto, le funzioni della RTE devono essere collegate ai punti di chiamata di CORSIM affinché la RTE risulti completamente funzionante. Per adempiere tale operazione è necessario utilizzare nuovamente lo strumento di configurazione cliccando sul pulsante per la modifica (i. e., *Edit*) di una RTE. A questo punto, selezionando la scheda relativa alle RTE, è possibile invocare lo strumento per effettuare il succitato collegamento. La figura 5.5 mostra il processo di collegamento tra le funzioni della RTE e i punti di chiamata di CORSIM.

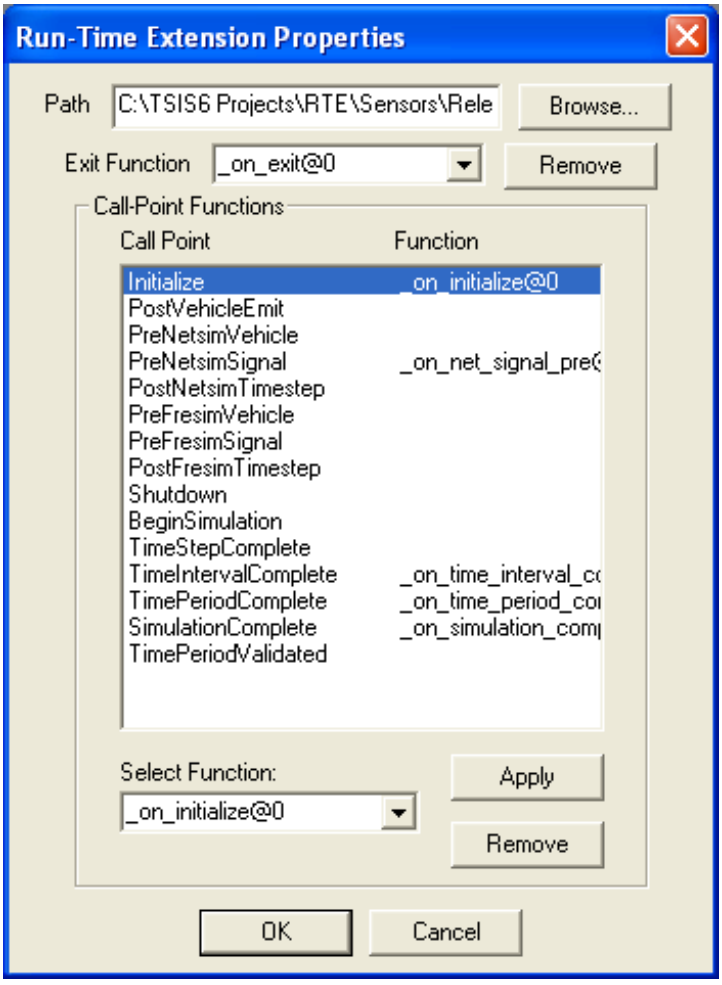
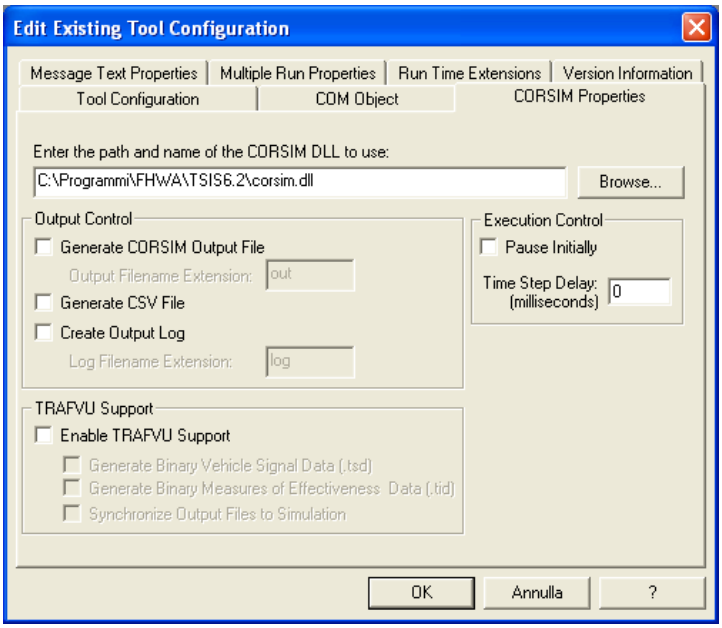


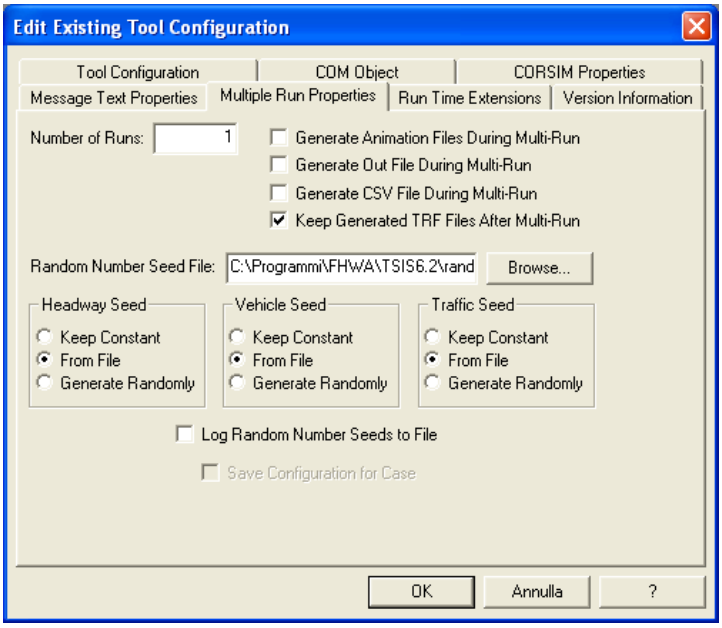
Figura 5.5: Collegamento delle funzioni della RTE ai rispettivi punti di chiamata di CORSIM.

Inoltre, può essere necessario dover configurare la RTE aggiunta in base alle sue esigenze, così come modificare alcune funzionalità di CORSIM relativamente ad essa. Ad esempio, la RTE per il monitoraggio e il tracciamento del passaggio dei veicoli sui sensori non necessita

che i file di output di CORSIM vengano generati, né che vengano generati i file per la visualizzazione animata della simulazione in TRAFED. Inoltre, tale RTE, non necessita che la simulazione CORSIM sia eseguita più volte. La figura 5.6 mostra la configurazione di tali opzioni.



(a) Proprietà di CORSIM.



(b) Impostazioni proprietà d'esecuzione di CORSIM.

Figura 5.6: Configurazione delle proprietà di CORSIM per la RTE: disattivazione dell'output di CORSIM, della generazione dell'input per TRAFED e delle esecuzioni multiple della simulazione.

5.2.5 Utilizzo delle API

Lungi dal volere fornire una documentazione esaustiva delle API dell'ambiente di sviluppo TSIS, in questa sezione, si intende presentare, tramite esempi, le modalità di utilizzo di tali API in linguaggio C++.

Ad esempio, il listato 5.1 mostra il codice di intestazione necessario a definire ed esportare la funzione `on_initialize` (listato 5.2), che, come mostrato dalla figura 5.5 a pagina 53 viene collegata al punto di chiamata `Initialize` di CORSIM. Si osservi che la scelta del nome di tale funzione non è vincolata ad alcun criterio.

```
1 #ifdef __cplusplus
2 #define DLL_EXPORT extern "C" __declspec(dllexport)
3 #endif // __cplusplus
```

Algoritmo 5.1: Costrutto per l'esportazione delle funzioni RTE

```
1 DLL_EXPORT void __stdcall on_initialize() {
2     // implementation
3 }
```

Algoritmo 5.2: Esempio di funzione RTE esportata

Il listato 5.3 illustra invece come importare una delle funzioni esposte dalla CORWIM API in una RTE. Nello specifico, l'esempio in questione, è relativo all'importazione della funzione `OutputString`, finalizzata all'invio di messaggi di output a TShell durante la simulazione CORSIM.

```
1 #ifdef __cplusplus
2 #define CORWINAPI extern "C" __declspec(dllimport)
3 CORWINAPI void __stdcall OutputString(const char *str,
    unsigned int size, int msgCode, unsigned long color);
4 #endif // __cplusplus
```

Algoritmo 5.3: Importazione delle CORWIN API

Infine, come già detto, CORSIM permette l'accesso alla maggior parte dei dati che esso manipola durante la simulazione, esportandoli nella memoria condivisa. Tali dati sono dei seguenti tipi:

1. variabili scalari (non array)
2. array allocati staticamente
3. array allocati dinamicamente
4. funzioni esposte tramite API.

Il listato 5.4 mostra il costrutto utilizzabile per l'importazione di dati e funzioni dalla cosiddetta CORSIM API.

```
1 #ifdef __cplusplus
2 #define DLL_IMPORT extern "C" __declspec(dllimport)
3 #endif // __cplusplus
```

Algoritmo 5.4: Costrutto per l'importazione delle CORSIM API

Tale costrutto viene poi utilizzato per l'effettiva importazione di dati e funzioni da CORSIM. Il listato 5.5 riporta un esempio di importazione per ogni tipo di dati che le CORSIM API rendono disponibile:

1. a linea 2 si importa il numero di sensori presenti sulla rete stradale urbana, esportato da CORSIM tramite la variabile scalare `NETSIM_DETECTORS_mp_NUMDET` e rinominato, a linea 3, in `net_det_num`

2. a linea 5 si importa la lista statica (i. e., di dimensione massima prefissata) dei numeri identificativi assegnati dall'utente ai nodi della rete stradale, `SIN075.NMAP`; rinominata a linea 9 in `net_node_num`

3. a linea 11 si importa la lista dinamica delle informazioni sui sensori, `*NETSIM_DETECTORS_mp_DETMOD`; rinominata, a linea 12, in `net_det_mod`

4. a linea 14 si importa `abortcorsim`, una funzione delle CORSIM API finalizzata al controllo dell'esecuzione della simulazione da parte della RTE

```
1  // netsim scalar non-array variables
2  DLL_IMPORT int NETSIM_DETECTORS_mp_NUMDET;
3  #define net_det_num NETSIM_DETECTORS_mp_NUMDET
4  // netsim statically allocated arrays
5  DLL_IMPORT struct
6  {
7      int NMAP[8999];
8  } SIN075;
9  #define net_node_num SIN075.NMAP
10 // netsim dinamically allocated arrays
11 DLL_IMPORT int *NETSIM_DETECTORS_mp_DTMODD;
12 #define net_det_id NETSIM_DETECTORS_mp_DTMOD
13 // netsim exported functions
14 DLL_IMPORT void __stdcall abortcorsim(void);
```

Algoritmo 5.5: Importazione di oggetti delle CORSIM API

5.3 ESTENSIONE

Avendo presentato l'ambiente di sviluppo TSIS e le API che esso fornisce per la sua estensione, è ora possibile descrivere l'estensione a tempo d'esecuzione sviluppata al fine di generare i succitati dataset.

Come detto nella sottosezione 5.1.2 a pagina 44, una delle principali limitazioni di CORSIM consiste nell'impossibilità di ottenere dei dati non aggregati dal processo di simulazione. Ciò poiché il minimo intervallo temporale che CORSIM permette di utilizzare è pari a 1 secondo per le reti stradali urbane e al più 0.1 secondi per le reti stradali extraurbane. Perciò, è emerso il problema di sorpassare tale limitazione. La RTE sviluppata, Sensors DLL, risponde a tale necessità monitorando determinati elementi (i. e., i sensori) di un qualsiasi tipo di rete stradale e tracciando gli eventi ad essi correlati (i. e., passaggio di un veicolo) su un file di output esterno a TSIS. Lo scopo di questa sezione consiste nel presentare il funzionamento di Sensors DLL.

5.3.1 Sensors DLL

L'obiettivo ultimo di Sensors DLL consiste nella generazione di un file CSV[33] che rappresenti il passaggio dei veicoli sui vari sensori presenti nella rete stradale nel tempo. Il monitoraggio dei sensori deve essere effettuato con la massima granularità temporale possibile (i. e., 0.1 secondi), anche nel caso di reti stradali urbane.

A tale scopo si è utilizzato il dato `NETSIM_DETECTORS_mp_DETON`, rinominato in `net_det_on`, esportato da CORSIM nella memoria condivisa. Tale campo indirizza un array dinamico la cui lunghezza è pari al numero di sensori sulla rete stradale. Ognuno degli elementi di tale array è costituito da 10 bit: l'i-esimo bit rappresenta l'attivazione (i. e., 1) o meno del relativo sensore nell'i-esimo passo temporale minimo (i. e., 0.1 secondi). Ogni elemento rappresenta perciò il passaggio dei veicoli sul sensore durante un intervallo temporale fisso di 1 secondo.

Di seguito si presentano le operazioni principali che Sensors DLL effettua:

1. ottenere il nome del file TRF di input, rappresentante la rete stradale e il modello di simulazione
2. effettuare il *parsing*[34] di tale file creando gli oggetti relativi a ogni elemento (e. g., intersezioni, strade, sensori) della rete stradale
3. rilevare il passaggio dei veicoli sui sensori ogni secondo

33 Comma Separated Values (CSV) è un formato basato su file di testo utilizzato per l'importazione ed esportazione (ad esempio da fogli elettronici o database) di una tabella di dati.

34 Il *parsing* consiste nel processo atto ad analizzare un input in modo da determinare la sua struttura grammaticale grazie ad una data grammatica formale.

4. ricostruire l'intero flusso di veicoli su ogni sensore durante tutto il tempo di simulazione

5. creare un file di output che contenga le informazioni ottenute.

Le operazioni 1 e 2 vengono compiute in corrispondenza dell'inizializzazione di CORSIM e quindi della RTE. Il risultato di tali operazioni è un insieme di istanze correlate rappresentanti gli elementi della rete stradale di input e le caratteristiche di ognuno di essi. Il diagramma delle classi di Sensors DLL, mostrato in figura 5.7 a pagina 60, illustra le relazioni di associazione e aggregazione degli oggetti con cui si è scelto di rappresentare le reti stradali TSIS. La classe `CNetwork` rappresenta la rete stradale, composta da un insieme di intersezioni e strade, elementi rappresentati rispettivamente dalle classi `CNode` e `CLink`. Ogni strada può a sua volta essere composta da più corsie, elementi rappresentati tramite la classe `CLane`, e contenere dei sensori, elementi rappresentati tramite la classe `CDetector`. Inoltre, poiché è possibile che alcuni sensori siano posti esclusivamente su una corsia piuttosto che su tutta la superficie della strada, sussiste una relazione anche fra la classe rappresentante le corsie e quella rappresentante i sensori. Invece, la classe `CBinary` non rappresenta alcun elemento concreto della rete stradale: la sua funzione è esclusivamente quella di incapsulare l'intero `net_det_on` e convertirlo nella corretta sequenza di bit rappresentante il flusso dei veicoli su un sensore. La procedura che effettua tali operazioni di inizializzazione della rete stradale nella RTE, chiamata `on_initialize`, è collegata al punto di chiamata `Initialize`, così come mostrato dalla figura 5.5 a pagina 53.

Configurata la rete stradale, Sensors DLL può monitorare i sensori di pari passo con l'esecuzione della simulazione da parte di CORSIM. Tale operazione viene effettuata ad ogni intervallo temporale di NETSIM poiché la procedura che la incorpora, chiamata `on_net_signal_pre`, è collegata al punto di chiamata `PreNetsimSignal` di CORSIM. Il listato 5.6 a pagina 59 illustra una versione semplificata del metodo C++ preposto all'esecuzione di tale procedura. Essa consiste nell'iterazione della lista di sensori afferenti ad una strada: per ogni sensore (ciclo a linea 8), recuperato l'identificatore che CORSIM utilizza per rappresentarlo (istruzione a linea 10), si ottiene il relativo elemento dell'array `net_det_on` (istruzione a linea 11), un intero rappresentante il flusso di veicoli sul sensore nell'ultimo secondo di simulazione. Tale intero viene poi convertito nella corretta sequenza di 10 bit tramite la classe `CBinary` a linea 12. Da linea 13 a linea 32 si itera in ordine inverso la sequenza di bit al fine di estrapolare e memorizzare lo stato (i. e., 1 in caso di veicolo rilevato, 0 altrimenti) del sensore in ogni passo temporale minimo (i. e., 0.1 secondi). Quindi questa procedura, ripetuta per tutte le strade presenti nella rete stradale e ad ogni intervallo temporale, memorizza per ogni sensore una lista di valori booleani.

Completata la simulazione e di conseguenza anche la procedura di monitoraggio dei sensori, Sensors DLL, in corrispondenza del punto di chiamata `SimulationComplete` di CORSIM, genera un file CSV in cui memorizza il tempo, il *time period* della rilevazione e la dinamica di stato di ogni sensore. La sottosezione 5.3.2 nella pagina seguente si occupa di presentare in maggior dettaglio l'output di Sensors DLL.

```
1  void CLink::processDetectors(void) {
2      int det, new_state, old_state  = 0;
3      POSITION pos, pos_i = NULL;
4      CDetector *detector = NULL;
5      CInteger *pi = NULL;
6      CBinarySequence *sequence = NULL;
7      pos = m_detector_list.GetHeadPosition();
8      while (pos != NULL) {
9          detector = m_detector_list.GetNext(pos);
10         int det_num = detector->getCorsimId();
11         det = net_det_on[det_num];
12         sequence = CBinarySequence::convert(det);
13         pos_i = sequence->sequence.GetTailPosition();
14         old_state = detector->getState() ? 1 : 0;
15         for (index = 0; index < 10; index++) {
16             pi = sequence->sequence.GetPrev(pos_i);
17             new_state = pi->data;
18             if ((old_state == 0) && (new_state == 1)) {
19                 detector->setState(true, !is_init);
20             }
21             if ((old_state == 1) && (new_state == 1)) {
22                 float atime = detector->getActivationTime();
23                 detector->setState(true, !is_init);
24             }
25             if ((old_state == 1) && (new_state == 0)) {
26                 detector->setState(false, !is_init);
27             }
28             if ((old_state == 0) && (new_state == 0)) {
29                 detector->setState(false, !is_init);
30             }
31             old_state = new_state;
32         }
33         delete sequence;
34     }
35 }
```

Algoritmo 5.6: Metodo della classe `CLink` per la rilevazione del passaggio dei veicoli sui sensori

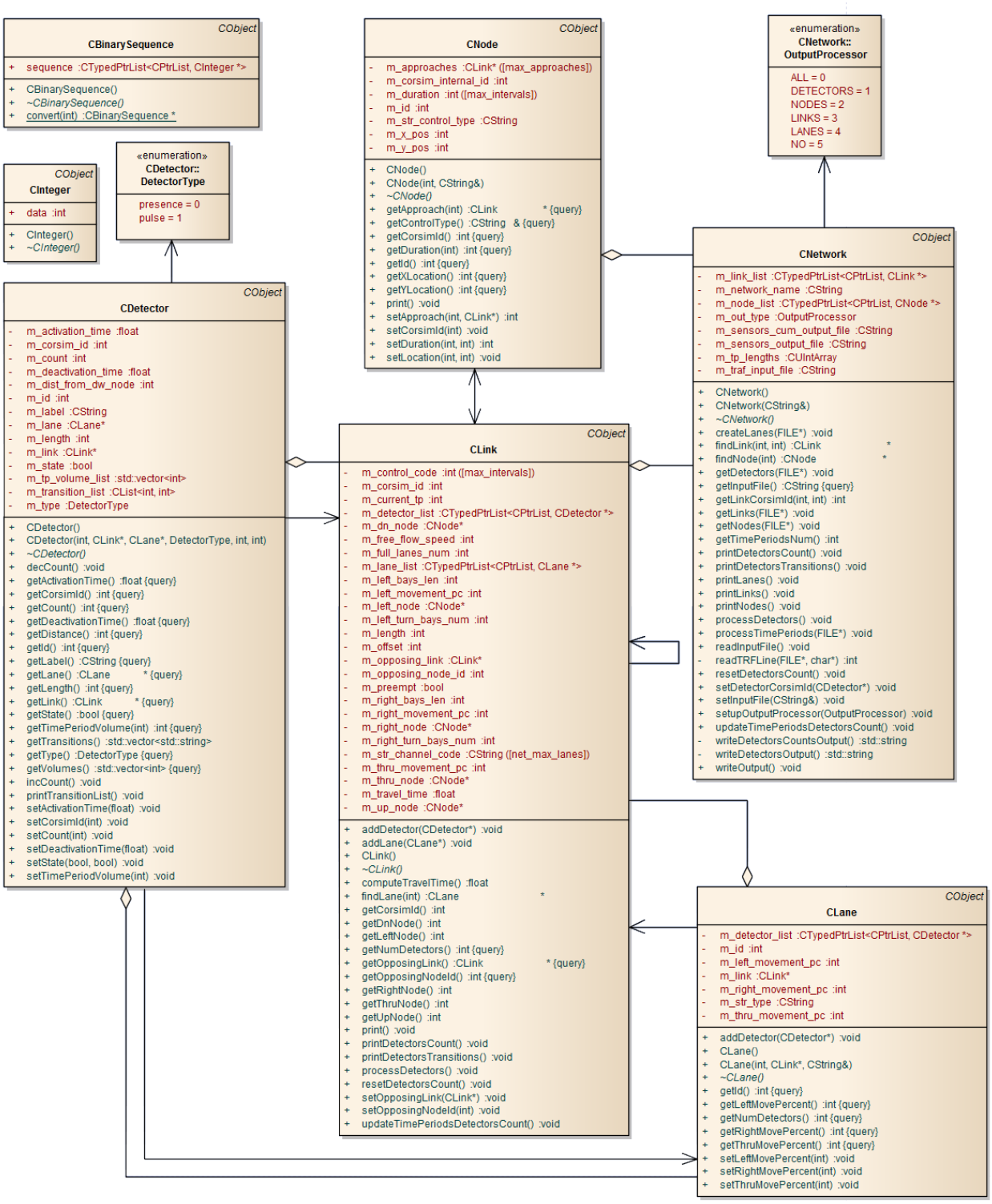


Figura 5.7: Diagramma delle classi di Sensors DLL.

5.3.2 Formato dell'output

Di seguito si presenta un esempio di file CSV di output generato da Sensors DLL.

```
time,tp,D131,D231,D232,D211,D212,D213
0.10000,1,0,0,0,0,0,0
0.20000,1,0,0,1,0,0,0
0.30000,1,0,0,1,0,0,0
0.40000,1,0,0,0,0,0,0
0.50000,1,0,0,0,0,1,0
0.60000,1,0,0,0,0,1,0
0.70000,1,0,0,0,0,0,0
0.80000,1,0,0,0,0,0,0
0.90000,1,0,0,0,0,0,0
1.00000,1,0,1,0,0,0,0
1.10000,2,0,1,0,0,0,1
1.20000,2,0,1,0,0,0,1
1.30000,2,0,1,0,0,0,0
1.40000,2,0,0,0,0,0,0
1.50000,2,0,0,1,0,0,0
1.60000,2,0,0,1,0,0,0
1.70000,2,0,0,0,0,0,0
1.80000,2,1,0,0,0,0,0
1.90000,2,1,0,0,0,0,0
2.00000,2,1,0,0,0,0,0
```

La tabella 5.3 chiarisce il significato di ogni colonna dei dati tabulari restituiti da Sensors DLL.

Nome colonna	**Descrizione**
time	Tempo di simulazione a cui è stato effettuato il monitoraggio dei sensori.
tp	Indice del corrente periodo temporale (i. e., *time period*).
identificatore sensore	1 in caso di presenza di un veicolo sul rispettivo sensore, 0 altrimenti.

Tabella 5.3: Descrizione della semantica dei file CSV generati da Sensors DLL.

Si osservi che, il fatto che il sensore `D232`, in corrispondenza dell'istante 1.6 (5ª colonna) abbia valore 1 non indica che il veicolo sia stato rilevato esattamente in tale istante, bensì ciò indica che durante l'intervallo temporale $[1.5, 1.6]$ tale sensore è stato attivato dal passaggio di un veicolo.

5.4 APPLICATIVI DI SUPPORTO

Al fine di automatizzare e facilitare la creazione di dataset relativi al traffico tramite la RTE oggetto del corrente capitolo, si è sviluppato

un insieme di strumenti dediti alla manipolazione dei file di output di Sensors DLL.

Di seguito si elencano le operazioni di manipolazione che tali strumenti supportano:

1. sostituzione (e eventualmente rimozione) della colonna relativa ai periodi temporali con una nuova colonna che rappresenti la classe di un insieme di osservazioni; tale colonna è automaticamente generata in base a un sistema di regole (e. g., *matching* tra periodo temporale e classe).

2. partizionamento del file di output di Sensors DLL in più file in base a vincoli temporali (e. g., divisione del file in blocchi di 60 secondi ciascuno)

3. ottimizzazione del file tramite rimozione delle linee duplicate (i. e., linee in cui non è avvenuto alcun cambiamento di stato dei sensori).

6 ESPERIMENTI NUMERICI

...

6.1 DATASET 1

...

6.1.1 Modello TSIS

...

6.1.2 Risultati

...

6.2 DATASET 2

...

6.2.1 Modello TSIS

...

6.2.2 Risultati

...

7 CONCLUSIONI

...

A GUIDE ALL'USO

A.1 UTILIZZO DEL PACKAGE CTBN

...

A.1.1 Caricamento del dataset

A.1.2 Calcolo delle sufficient statistics

A.1.3 Calcolo dei parametri

A.1.4 Calcolo delle CIM

A.1.5 Apprendimento

A.1.6 Classificazione

A.1.7 Apprendimento strutturale

A.1.8 Cross-validation

A.2 CREAZIONE DI DATASET

...

A.2.1 Sensors DLL

...

Installazione

...

Guida all'uso

...

A.2.2 Applicativi di supporto

...

ACRONIMI

API	Application Programming Interface 48
BN	Bayesian Network 1
BNC	Bayesian Network classifier 22
CIM	Conditional Intensity Matrix 10
CORSIM	Corridor microscopic simulation program 43
COM	Component Object Model 48
CPD	Conditional Probability Distribution 2
CPT	Conditional Probability Table 2
CSV	Comma Separated Values 57
CTBN	Continuos time Bayesian Network 1
CTBNC	Continuos time Bayesian Network classifier 21
CTNB	Continuos time Naive Bayes 21
CTNBC	Continuos time Naive Bayes classifier 22
CTTANB	Continuos time tree augumented Naive Bayes 21
CTTANBC	Continuos time tree augumented Naive Bayes classifier 22
DAG	Directed acyclic graph 1
DBN	Dynamic Bayesian Networks 33
DLL	Dynamic-link library 43
EM	Expectation Maximization 4
FRESIM	Freeway Simulator 43
FWHA	Federal Highway Administration 41
GUI	Graphical User Interface 42
IDE	Integrated Development Environment 41
IC	Inductive Causation 5
IM	Intensity Matrix 7
MAP	Maximum a posteriori 28
MCMC	Markov Chain Monte Carlo 5
MLE	Maximum Likelihood Estimation 18
NETSIM	Network Simulator 43
PV	Process Variable 11
RT	Record Type 42
RTE	Run-Time Extension 43
TAN	Tree Augumented Naive Bayes 22
TNO	TRAFED Native Object 42
TRAF	Traffic File 42
TRAFED	TRAF Editor 43
TRAFVU	TRAF Visualization Utility 44
TRF	Traffic File 42

TShell	TSIS Shell 43
TSIS	Traffic Software Integrated System 41
VBScript	Microsoft's Visual Basic Scripting Edition 44

INDICE ANALITICO

BIBLIOGRAFIA

Chickering, David Maxwell

1994 *Learning Bayesian networks is NP-hard*, rapp. tecn., Microsoft Research. (Citato alle p. 33, 37.)

2013 «A Transformational Characterization of Equivalent Bayesian Network Structures», *CoRR*, p. 87-98, http://arxiv.org/abs/1302.4938. (Citato a p. 6.)

Chickering, David Maxwell, David Heckerman e Christopher Meek

2004 «Large-sample learning of Bayesian networks is NP-hard», *... Journal of Machine Learning ...*, 5, p. 1287-1330, http://dl.acm.org/citation.cfm?id=1044703. (Citato a p. 33.)

Dempster, A P, N M Laird e D B Rubin

1977 «Maximum likelihood from incomplete data via the EM algorithm», *Journal of the Royal Statistical Society Series B Methodological*, Series B, 39, 1, p. 1-38, ISSN: 00359246, DOI: 10.2307/2984875, http://www.jstor.org/stable/2984875. (Citato a p. 4.)

Duda, R. O. e P. E. Hart

1973 *Pattern Classification and Scene Analysis*, John Willey & Sons, New Yotk. (Citato a p. 21.)

Friedman, N, D Geiger e M Goldszmidt

1997 «Bayesian network classifiers», *Machine learning*, 163, p. 131-163, http://link.springer.com/article/10.1023/A:1007465528199. (Citato alle p. 21, 22, 29.)

Geman, Stuart e Donald Geman

1984 «Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images», *IEEE Trans. Pattern Anal. Mach. Intell.*, 6, 6, p. 721-741, ISSN: 0162-8828, DOI: 10.1109/TPAMI.1984.4767596, http://dx.doi.org/10.1109/TPAMI.1984.4767596. (Citato a p. 5.)

Gihman, Iosif I. e Anatolij V. Skorohod

1973 *The theory of stochastic processes II*, New York: Springer-Verlag. (Citato a p. 9.)

Gilks, WR, S Richardson e DJ Spiegelhalter

1996 «Markov chain Monte Carlo in practice». (Citato a p. 5.)

Heckerman, David

1996 «A Tutorial on Learning With Bayesian Networks», *Innovations in Bayesian Networks*, Studies in Computational Intelligence, 1995, November, a cura di Dawn E Holmes e Lakhmi C Jain, p. 33-82, ISSN: 1860949X, DOI: 10.1007/978-3-540-85066-3, http://www.springerlink.com/index/62mv333389016034.pdf. (Citato alle p. 5, 18, 19.)

Heckerman, David, Dan Geiger e David M. Chickering

1995 «Learning Bayesian networks: The combination of knowledge and statistical data», *Machine Learning*, 20, 3 (set. 1995), p. 197-243, ISSN: 0885-6125, DOI: 10.1007/BF00994016, http://link.springer.com/10.1007/BF00994016. (Citato alle p. 6, 19.)

Korb, K.B. e A.E. Nicholson

2011 *Bayesian Artificial Intelligence*, Chapman & Hall / CRC Computer Science and Data Analysis, CRC PressINC, ISBN: 9781439815915. (Citato alle p. 1, 3.)

Langley, Pat, Wayne Iba e Kevin Thompson

1992 «An Analysis of Bayesian Classifiers», in *AAAI*, p. 223-228, ISBN: 0-262-51063-4. (Citato alle p. 21, 23.)

Loève, Michel

1978 *Probability theory. II Edition.* Fourth, Graduate Texts in Mathematics, Vol. 46, Springer-Verlag, New York, p. xvi+413, ISBN: 0-387-90262-7. (Citato alle p. 6, 7.)

MacKay, D. J. C.

1998 «Introduction to Monte Carlo methods», in *Proceedings of the NATO Advanced Study Institute on Learning in graphical models*, Kluwer Academic Publishers, Norwell, MA, USA, p. 175-204, http://dl.acm.org/citation.cfm?id=299068.299077. (Citato a p. 5.)

Nodelman, Uri D.

2007 *Continuos Time Bayesian Networks*, tesi di dott., Stanford University. (Citato alle p. 9, 11, 12, 16, 18.)

Nodelman, Uri D., CR Shelton e Daphne Koller

2002 «Learning continuous time Bayesian networks», *Proceedings of the Nineteenth ...*, X, arXiv: /arxiv.org/abs/1212.2498[http:], http://dl.acm.org/citation.cfm?id=2100639. (Citato alle p. 15, 33, 37.)

Norris, James R.

1998 *Markov chains*, Cambridge series in statistical and probabilistic mathematics, Cambridge University Press, p. I-XVI, 1-237, ISBN: 978-0-521-48181-6. (Citato a p. 7.)

Pearl, Judea

1988 *Probabilistic reasoning in intelligent systems: networks of plausible inference*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ISBN: 0-934613-73-7. (Citato a p. 5.)

Russell, Stuart J. e Peter Norvig

2003 *Artificial Intelligence: A Modern Approach*, Pearson Education, ISBN: 0137903952, http://portal.acm.org/citation.cfm?id=773294. (Citato alle p. 2, 3, 37, 38.)

Shachter, Ross D. e Mark A. Peot

1990 «Simulation Approaches to General Probabilistic Inference on Belief Networks», in *Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence*, UAI '89, North-Holland Publishing Co., Amsterdam, The Netherlands, p. 221-234, ISBN: 0-444-88738-5, http://dl.acm.org/citation.cfm?id=647232.719570. (Citato a p. 5.)

Steck, Harald and Jaakkola, Tommi S

2002 «On the Dirichlet Prior and Bayesian Regularization», *Advances in Neural Information Processing Systems*, September, p. 713-720. (Citato a p. 19.)

Stella, F e Y Amer

2012 «Continuous time Bayesian network classifiers.» *Journal of biomedical informatics*, 45, 6 (dic. 2012), p. 1108-19, ISSN: 1532-0480, DOI: 10.1016/j.jbi.2012.07.002, http://www.ncbi.nlm.nih.gov/pubmed/22846170. (Citato alle p. 10, 12, 22, 25, 28, 31.)

Verma, Thomas S. e Judea Pearl

1991 «Equivalence and synthesis of causal models», in *Uncertainty in Artificial Intelligence*, North Holland, p. 255-268. (Citato a p. 5.)

DICHIARAZIONE

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices.

Milano, settembre 2013

Leonardo Di Donato