

SEMINARIO DEL 04/03

Ruggieri Andrea

Stranieri Francesco

MAD Lab



REFERENZE

Understanding Bayesian Networks with Examples in R

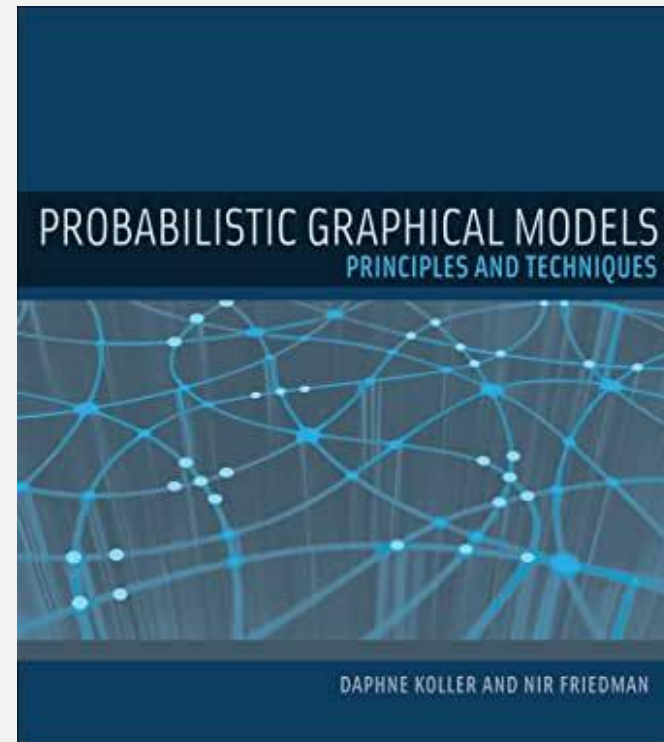


UNIVERSITY OF
OXFORD

Marco Scutari

scutari@stats.ox.ac.uk
Department of Statistics
University of Oxford

January 23–25, 2017



Introduzione

Reti Bayesiane

GRAFO E DISTRIBUZIONE DI PROBABILITÀ

Le Reti Bayesiane (BN) sono definite da:

- una **struttura di rete**, nello specifico da un **DAG** (grafo aciclico diretto) $\mathcal{G} = (V, A)$ nel quale ogni nodo $v_i \in V$ corrisponde ad una variabile casuale X_i ;
- una **distribuzione di probabilità globale** X con parametri θ , la quale può essere scomposta in **distribuzioni di probabilità locali** più piccole, secondo gli archi $a_{ij} \in A$ presenti nel grafo \mathcal{G} .

Il ruolo principale della struttura di rete consiste nell'esprimere le relazioni di **indipendenza condizionale** tra le variabili del modello attraverso la **separazione grafica**, specificando così la fattorizzazione della distribuzione globale:

$$P(X) = \prod_{i=1}^N P(X_i | \Pi_{X_i}; \theta_{X_i}) \text{ dove } \Pi_{X_i} = \{\text{genitori di } X_i\}$$

DISTRIBUZIONI DI PROBABILITÀ

La distribuzione di probabilità $P(\mathbf{X})$ dovrebbe essere scelta in modo tale che la BN:

- possa **apprendere efficientemente** dai dati;
- sia **flessibile** (le ipotesi sulla distribuzione non dovrebbero essere troppo restrittive);
- sia **facile da interrogare** per compiere inferenza.

Le tre scelte più comuni in letteratura sono:

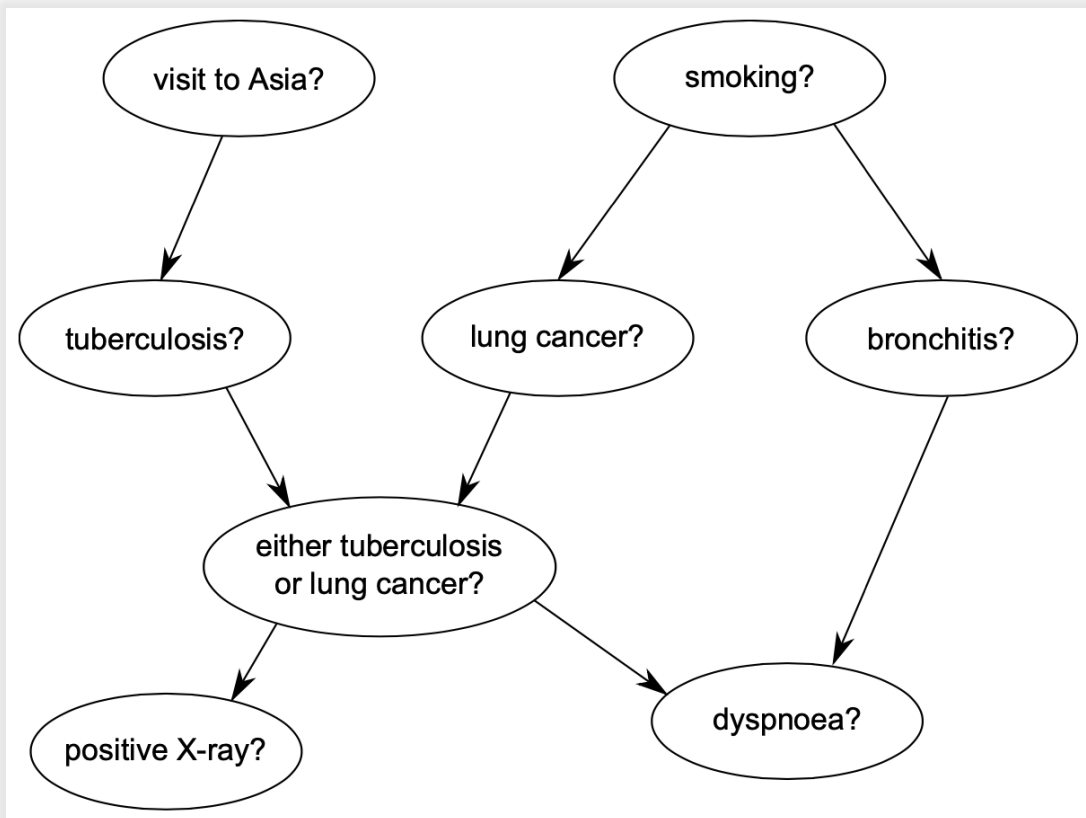
1. BN **discrete** (DBN), nelle quali \mathbf{X} e le $X_i | \prod_{X_i}$ sono distribuzioni multinomiali;
2. BN **Gaussiane** (GBN);
3. BN **lineari Gaussiane condizionali** (CLGBN).

La loro popolarità è data dal fatto che in letteratura è stato dimostrato che in questi tre casi è **possibile eseguire un'inferenza esatta**.

BN DISCRETE (DBN)

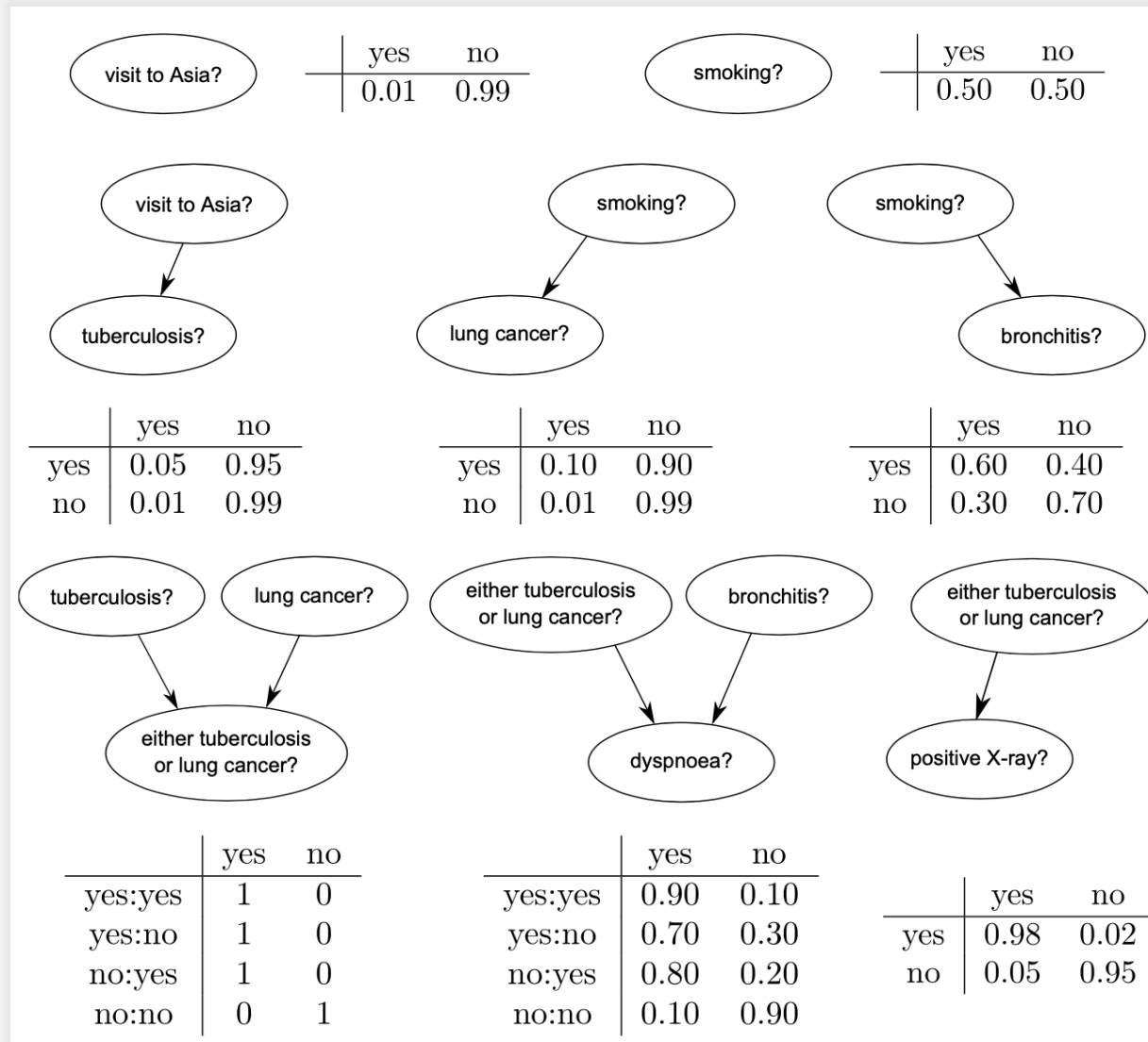
Un esempio classico di DBN è il network **ASIA** [Lauritzen & Spiegelhalter, 1998] che include una raccolta di variabili binarie.

La rete descrive un semplice problema per la diagnosi della tubercolosi e del cancro ai polmoni.



Parametri totali di X :
 $2^8 - 1 = 255$

TABELLA DI PROBABILITÀ CONDIZIONATA (CPT)



Le distribuzioni locali $X_i | \prod_{X_i}$ assumono la forma di **tabelle di probabilità condizionata** per ogni nodo, date tutte le possibili combinazioni dei valori dei propri genitori.

Parametri totali delle $X_i | \prod_{X_i}$: 18

Fondamenti di Structure Learning

Algoritmi di apprendimento Score-based

ALGORITMI DI APPRENDIMENTO STRUTTURATO

Obiettivo: imparare la struttura del grafo della rete bayesiana sulla base dei dati osservati $P(\mathcal{G} | D)$.

Esistono diverse classi di algoritmi per apprendere la struttura di una rete bayesiana:

- **Algoritmi constraint-based:** si usano test statistici per individuare le relazioni di indipendenza condizionale dei dati;
- **Algoritmi Score-based:** viene associato uno score (punteggio) a ogni DAG candidato;
- **Algoritmi ibridi :** vengono alternate fasi di restrizione dei possibili DAG e fasi di massimizzazione, secondo lo score.

Il problema dell'apprendimento strutturato può essere visto come un problema di **ottimizzazione** su uno spazio combinatorio di tutte le strutture di grafi possibili. Solitamente questo è un problema **NP-hard** e per questo sono state introdotti **metodi euristici**.

N.B.: in questa prima parte soffermeremo l'attenzione esclusivamente sugli **algoritmi Score-based** e in particolare su Hill-Climbing (*hc*) e Tabu search (*tabu*).

HILL-CLIMBING ALGORITHM

Si tratta di un **metodo euristico**:

- orientato alla ricerca di una buona soluzione che non è necessariamente l'ottima;

In particolare, data una grande quantità di dati e una **funzione euristica**, l'algoritmo prova a cercare una soluzione sufficientemente buona al problema:

- HC si basa su 3 operazioni diverse sul grafo: **aggiunta** di un arco, **cancellazione** di un arco oppure **reverse** di un arco;

Hill-Climbing ha due caratteristiche:

- è una variante dell'**algoritmo di generazione e di test**, in quanto il feedback viene ottenuto dalla procedura di test. Questo viene poi utilizzato dal generatore per decidere la mossa successiva nello spazio di ricerca;
- usa un **approccio greedy**;

Sono stati esaminati due tipi di Hill-Climbing Algorithms:

- **Simple Hill-Climbing**: esamina i nodi vicini uno per uno e seleziona il primo nodo vicino che ottimizza il costo corrente, assumendolo come nodo successivo. Si tratta di un algoritmo veloce ma la soluzione potrebbe essere non ottimale oppure non garantita;
- **Stochastic Hill-Climbing**: non esamina tutti i nodi vicini prima di decidere quale nodo selezionare. Si limita a selezionare un nodo vicino in maniera randomica e decide, sulla base del miglioramento osservato, se spostarsi in quel nodo vicino o se esaminarne un altro.

<https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/>

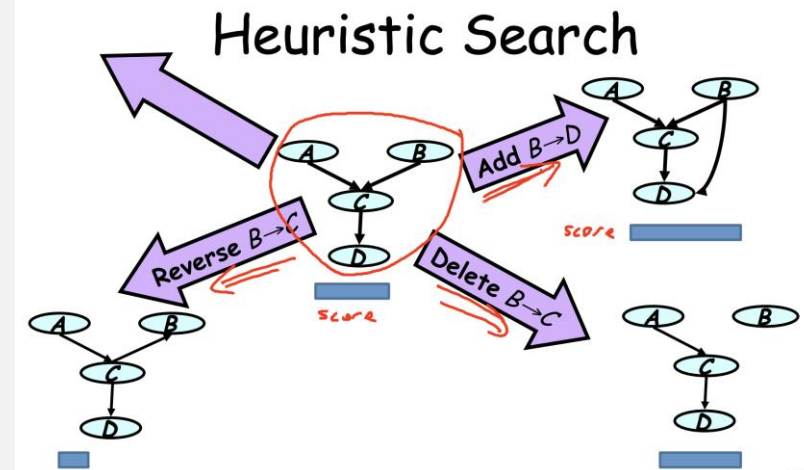
<https://www.javatpoint.com/hill-climbing-algorithm-in-ai>

<https://medium.com/@bhavek.mahyavanshi50/introduction-to-hill-climbing-artificial-intelligence-a3714ed2d8d8>



HILL-CLIMBING ALGORITHM

1. Scegli una rete \mathcal{G} iniziale (solitamente viene scelta vuota ma non è obbligatorio).
2. Ad ogni iterazione calcola lo score di \mathcal{G} (denotato come $Score_{\mathcal{G}} = Score(\mathcal{G})$).
3. Metti $maxscore = Score_{\mathcal{G}}$.
4. Ripeti i seguenti step finché $maxscore$ aumenta:
 - I. Per ogni possibile aggiunta, eliminazione o inversione di un arco che non porta a un ciclo
 - a) Calcola lo score della rete modificata \mathcal{G}^* , $Score_{\mathcal{G}^*} = Score(\mathcal{G}^*)$.
 - b) Se $Score_{\mathcal{G}^*} > Score_{\mathcal{G}}$ allora imposta $\mathcal{G} = \mathcal{G}^*$ e $Score_{\mathcal{G}} = Score_{\mathcal{G}^*}$.
 - II. Aggiorna $maxscore$ con il nuovo valore di $Score_{\mathcal{G}}$.
5. Ritorna il DAG ottenuto.



Daphne Koller

TABU SEARCH (1)

- Come Hill-Climbing si tratta di un **metodo euristico**.
- Memorizza le informazioni riguardanti il processo di ricerca, in modo da uscire efficientemente da situazioni di stallo.
- Gli attributi sono informazioni sintetiche che devono consentire di evitare il passaggio per soluzioni già esaminate.
- Gli attributi vengono memorizzati in una o più **liste TABU**. In particolare, ad ogni iterazione vengono aggiunti nuovi attributi.
- Le liste TABU hanno una dimensione limitata e adottano una politica **FIFO**. Fissando TL la lunghezza di una lista TABU, un attributo che entra in lista all'iterazione k , vi esce all'iterazione $k + TL$.
- La lunghezza TL di una lista può essere fissata in due modi: attraverso **regole statiche** o **regole dinamiche**.

<http://www.federica.unina.it/ingegneria/ricerca-operativa-ing-2/tabu-search/>
<https://www.sciencedirect.com/topics/computer-science/tabu-search>

TABU SEARCH (2)

Tabu search si basa su 3 passi principali, riassunti come segue:

- **inizializzazione:** si considera S una soluzione di innesco dell'algoritmo;
- **definizione di intorno $N(S, k)$ e scelta della nuova soluzione S' :** si definisce l'intorno della soluzione corrente e si sceglie una nuova soluzione S' sulla base di uno stimatore $E[N(S, k)]$. Si sostituisce S con S' ;
- **criterio di arresto.**

N.B.: l'inizializzazione dell'algoritmo può essere casuale oppure dovuta da un algoritmo costruttivo. Lo stimatore solitamente coincide con la funzione obiettivo.

Per migliorare la qualità di una soluzione ottenuta ci sono **2 fasi**:

- **intensificazione:** la ricerca di nuove soluzioni si concentra su intorni di soluzioni buone già individuate (exploitation);
- **diversificazione:** si sposta la ricerca delle nuove soluzioni verso nuove regioni che ancora non sono state esplorate (exploration). Solitamente la diversificazione richiede di modificare sensibilmente la soluzione corrente o la migliore soluzione ottenuta;

STRUCTURED LEARNING - CONCLUSIONI

Gli algoritmi di apprendimento strutturato sono utili per costruire modelli predittivi quando:

- gli **esperti di dominio** non conoscono esattamente la struttura della rete;
- per scoprire **nuova conoscenza**.

Tuttavia, trovare lo score più alto è un problema **NP-hard**.

Tipicamente, il problema viene risolto attraverso l'uso di semplici **euristiche** basate su:

- **Local search**: aggiunta di un arco, eliminazione di un arco, reverse di un arco;
- **Hill-Climbing** con liste TABU e random restarts;
- altri algoritmi migliori.

Inferenza Causale

Classi di dati mancanti, Missingness Mechanism

DATI MANCANTI

Le variabili latenti sono solo un tipo di dati mancanti:

- una variabile **latente** è una variabile **di cui non sappiamo nulla**, né la sua posizione nella BN né la sua distribuzione;
- una variabile **non osservata** è una variabile **che non osserviamo**, ma di cui conosciamo la posizione e la distribuzione;
- una variabile **parzialmente osservata** è una variabile di cui **osserviamo soltanto alcuni campioni**, non tutti (i restanti sono denotati come *NA*).

I **problemi principali** che sorgono dalla mancanza dei dati sono:

- come apprendiamo **la struttura della BN dai dati**?
- dato un DAG, come stimiamo **i parametri delle distribuzioni locali**?

La risposta a entrambe le domande sono gli algoritmi Expectation-Maximization e Data Augmentation.

CLASSI DI DATI MANCANTI

Esistono **tre classi di dati mancanti**:

- completamente mancanti at random (**MCAR**) - **Non esiste relazione** tra la mancanza dei dati e nessun valore, osservato o mancante. Questi dati mancanti sono un sottoinsieme casuale dei dati.
- mancanti at random (**MAR**) - **Esiste una relazione sistematica** tra la propensione dei dati mancanti e i dati osservati, ma non con i dati mancanti.
- mancanti non at random (**MNAR**) - **Esiste una relazione tra la propensione** di un valore ad essere mancante e il suo valore.

MNAR sono non-ignorabili dato che il meccanismo dei dati mancanti stesso deve essere modellato (perché i dati sono mancanti e si conosce quale sia il loro valore probabile).

MCAR e MAR sono entrambe considerate ignorabili dato che non si deve includere nessuna informazione a proposito dei dati mancanti in sé quando abbiamo a che fare con i dati mancanti.

RAPPRESENTAZIONE DEL MISSINGNESS MECHANISM

- Nel contesto delle BNs, ogni variabile ha una distribuzione locale $X_i \sim P(X_i | \Pi_{X_i})$ se i dati sono completi.
- Se X_i ha dati mancanti, nel caso di **MAR** e **MCAR**:

$$X_i \sim \begin{cases} P(X_i | \Pi_{X_i}) & \text{per dati osservati } X_i^{(O)} \\ P(X_i | \Pi_{X_i}) & \text{per dati mancanti } X_i^{(M)} \end{cases}$$

- Mentre, nel caso di **MNAR**:

$$X_i \sim \begin{cases} P(X_i^{(O)} | \Pi_{X_i}) & \text{per dati osservati } X_i^{(O)} \\ P(X_i^{(M)} | \Pi_{X_i}) & \text{per dati mancanti } X_i^{(M)} \end{cases}$$

dove **M** è il missing mechanism.

- **M** è non-ignorabile dato che non può essere propriamente stimato dalla distribuzione locale.

IMPUTAZIONE DI DATI MANCANTI (1)

L'imputazione di valori mancanti in un data set incompleto implica:

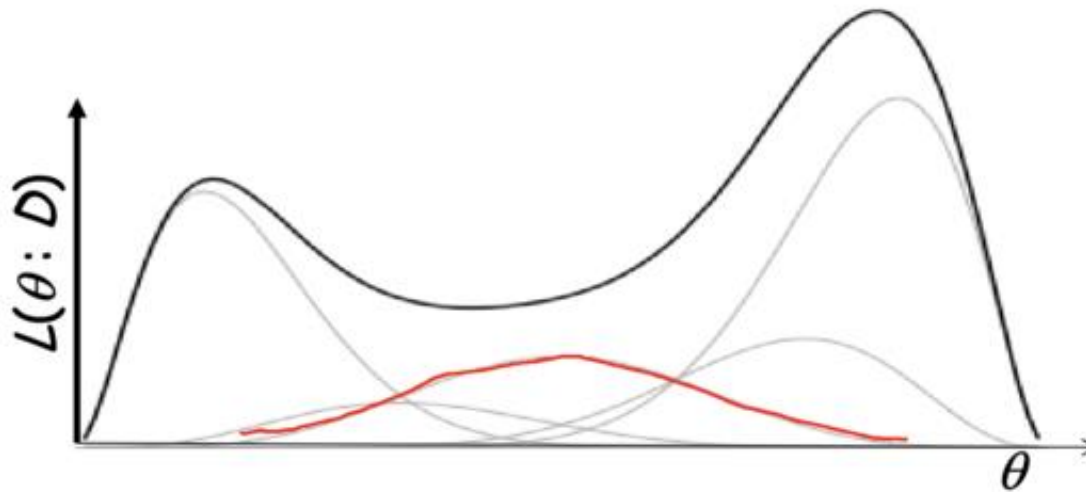
- sostituirli con le loro **posterior expectations o maximum a posteriori** stimata in un **contesto bayesiano**;
- sostituirli con la loro **maximum likelihood** stimata usando i loro genitori, in un **contesto frequentista**.

In entrambi i casi:

- si necessita di una **BN completamente specificata**;
- risulta preferibile apprendere la BN in un modo bayesiano/frequentista per eseguire imputazioni in modo bayesiano/frequentista;
- tutte le informazioni necessarie per fare inferenza su ogni nodo sono incluse nel suo **Markov blanket**, quindi non è necessario il resto della rete per imputare valori mancanti per quel nodo.

IMPUTAZIONE DI DATI MANCANTI (2)

Multimodal Likelihood



Daphne Koller

Algoritmo Expectation- Maximization

Algoritmo EM, Proprietà di EM, Structural EM

PASSI DELL'ALGORITMO EM

L'algoritmo Expectation-Maximization è suddiviso nei seguenti passi:

- scegliere un valore iniziale $\hat{\theta}_0$ per θ .
- fino a che $|\hat{\theta}_{j-1} - \hat{\theta}_j| < \epsilon$ e j incrementale, ripetere:
 - **Expectation Step** – Calcolare la **distribuzione di probabilità dei dati mancanti**:

$$P\left(X_i^{(M)} | X_i^{(O)}, \hat{\theta}_j\right) = \frac{P\left(X_i^{(O)} | X_i^{(M)}, \hat{\theta}_j\right) P\left(X_i^{(M)} | \hat{\theta}_j\right)}{\int P\left(X_i^{(O)} | X_i^{(M)}, \hat{\theta}_j\right) P\left(X_i^{(M)} | \hat{\theta}_j\right)}$$

- **Maximization Step** – Calcolare la **nuova stima** $\hat{\theta}_j$ dato $P\left(X_i^{(M)} | X_i^{(O)}, \hat{\theta}_j\right)$.
- stimare θ con l'ultimo $\hat{\theta}_j$.

PROPRIETÀ DELL'ALGORITMO EM

Esistono **implementazioni di EM** sia frequentistiche che bayesiane, la prima stima maximum likelihood mentre la seconda stima maximum posterior.

EM è garantito che converga ma:

- potrebbe convergere ad un **massimo locale**;
- la convergenza potrebbe essere arbitrariamente **lenta**.

Per le BNs, la convergenza è garantita solo se tutti i passaggi vengono portati avanti con **inferenza esatta**. La variabilità aggiunta dall'inferenza approssimata può far deragliare la convergenza.

ESTENSIONE DELL'ALGORITMO EM – STRUCTURAL EM

Apprendere il (CP)DAG di una BN in presenza di dati mancanti (in aggiunta ai parametri) è un problema che rappresenta una sfida da un punto di vista sia statistico che computazionale.

Friedman estese l'algoritmo EM per lavorare su questa task e chiamò il risultante algoritmo **Structural EM**:

- inizializza BN \mathcal{B}_0 con un DAG vuoto G_0 (dunque senza archi).
- fino a che \mathcal{B}_i è diverso da \mathcal{B}_{i-1} :
 - **Expectation Step** – Imputare i dati mancanti con le loro posterior expectations o la loro maximum likelihood stimata usando la BN corrente.
 - **Maximization Step** – Apprendere una BN aggiornata dai dati completati.

Pratica: R

Script EM manuale

EM MANUALE (1)

Approccio metodologico:

- Intero script realizzato su R. Funzioni di **expectation** e di **maximisation** sono scritte a mano senza utilizzare funzioni presenti in *bnlearn*;
- I dati mancanti vengono aggiornati ad ogni iterazione sulla base dell'expectation step;
- Si tratta di un approccio **top-down**: si parte da un esempio e si cerca di generalizzare l'algoritmo su diversi tipi di strutture;
- Tutti gli step sono eseguiti attraverso **inferenza esatta**;
- Esempio iniziale preso dalle slides «*Understanding Bayesian Networks*» di **Marco Scutari**;
- Facilità di confrontare i risultati per capire se la procedura risulta essere corretta.

Principali problemi riscontrati:

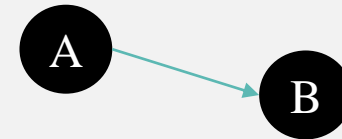
- Inizialmente non tutti gli step e le formule erano chiare. E' stato richiesto uno studio attento dell'argomento;
- Nonostante i risultati ottenuti siano uguali alle slide, si vedrà che i risultati potrebbero essere diversi dai risultati ottenuti tramite la libreria *bnlearn*. Nasce un problema di **interpretazione dei risultati**;
- L'implementazione degli step di expectation e di maximisation richiede molta attenzione e diversi **debug**.

EM MANUALE (2)

Approccio metodologico

- L'implementazione dell'algoritmo EM manuale parte con dai seguenti dati e dalla seguente struttura:

	Data_1	Data_2	Data_3	Data_4	Data_5	Data_6	Data_7	Data_8	Data_9	Data_10
A	0	0	0	NA	NA	NA	1	1	1	1
B	0	1	1	1	0	0	0	0	1	NA



- Si inizializzano le CPT dei nodi A e B e si iterano i seguenti step:
 - Expectation** – calcolo delle probabilità a posteriori dei dati mancanti;
 - Updating** – assegnazione di un valore binario ai dati mancanti sulla base delle probabilità a posteriori appena calcolate (step non necessario ma utile per comprendere i dati rimpiazzati);
 - Maximisation** - aggiornamento delle nuove CPT sulla base del passo di Expectation.
- Vengono confrontati i risultati ottenuti.
- Sulla base dei risultati ottenuti si è cercato di generalizzare il più possibile lo script ad altri tipi di strutture bayesiane, assumendo tutte le variabili come **binarie**.

$$\pi = \frac{1}{n} \sum_{x_i} \mathbb{1}_O + \mathbb{1}_M \pi_{x_i}^M$$

EM MANUALE (3)

Inizializzazione

- Siccome inizialmente c'è massima incertezza, si sostituisce il valore NA nei dati mancanti con 0.5.

(1) Dati iniziali

	Data_1	Data_2	Data_3	Data_4	Data_5	Data_6	Data_7	Data_8	Data_9	Data_10
A	0	0	0	0.5	0.5	0.5	1	1	1	1.0
B	0	1	1	1.0	0.0	0.0	0	0	1	0.5

- La CPT di A viene calcolata assumendo entrambi i valori 0/1 come equiprobabili. La CPT di B viene ottenuta attraverso la semplice computazione della probabilità condizionata.
- La matrice dei **missing probabilities** conterà 0 se la probabilità della mancanza del dato è uguale a 0 (dato osservato). Conterà 1 se quel dato risulta essere missing.

(2) Missing probabilities

	Data_1	Data_2	Data_3	Data_4	Data_5	Data_6	Data_7	Data_8	Data_9	Data_10
A	0	0	0	1	1	1	0	0	0	0
B	0	0	0	0	0	0	0	0	0	1

N.B.: nello script R, gli 0 sono stati sostituiti con il testo *'observed'* per indicare che quel dato è osservato.

```
> cpt
$A
A
 0  1
0.5 0.5

$B
A
B      0      1
0 0.3333333 0.6666667
1 0.6666667 0.3333333
```

N.B.: è importante distinguere la matrice dei dati osservati (1) con la matrice della probabilità dei missing (2). La prima matrice ha lo scopo di riportare i dati osservati e i dati rimpiazzati dopo l'applicazione dell'algoritmo EM. La seconda matrice ha lo scopo di memorizzare le probabilità a posteriori dei missing values, calcolati nello step di expectation.

EM MANUALE (4)

Iterazione 1

Ad ogni iterazione vengono eseguiti i seguenti passi:

- **Expectation:** vengono calcolate le probabilità a posteriori dei dati mancanti.

	Data_1	Data_2	Data_3	Data_4	Data_5	Data_6	Data_7	Data_8	Data_9	Data_10
A	0	0	0	0.6666667	0.3333333	0.3333333	0	0	0	0.0000000
B	0	0	0	0.0000000	0.0000000	0.0000000	0	0	0	0.6666667

P(A=0|B=1)

- **Updating:** i dati mancanti vengono aggiornati attraverso un'assegnazione di un valore binario 0/1.

	Data_1	Data_2	Data_3	Data_4	Data_5	Data_6	Data_7	Data_8	Data_9	Data_10
A	0	0	0	0	1	1	1	1	1	1
B	0	1	1	1	0	0	0	0	1	0

- **Maximisation:** vengono aggiornate le Conditional Probability Tables.

$$\pi = \frac{1}{n} \sum_{x_i} \mathbb{1}_O + \mathbb{1}_M \pi_{x_i}^M$$

```

$A
A
      0      1
0.4333333 0.5666667

$B
A
      0      1
B 0 0.3846154 0.7058824
   1 0.6153846 0.2941176
    
```

EM MANUALE (5)

- All'**iterazione 2** si ricavano i seguenti valori:

- **Expectation step:**

(dato 4) $P(A=0|B=1) = 0.615$

(dato 5, 6) $P(A=0|B=0) = 0.294$

(dato 10) $P(B=0|A=1) = 0.706$

- **Maximisation step:**

CPT A $P(A=0) = 0.4203$

$P(A=1) = 0.5796$

CPT B $P(B=0|A=0) = 0.3778$

$P(B=0|A=1) = 0.7103$

$P(B=1|A=0) = 0.6221$

$P(B=1|A=1) = 0.2896$

- Vengono eseguite n iterazioni (in questo caso $n = 10$)

```
for (i in 1:10) {  
  M.prob = expectation_step()  
  Data = update_data()  
  cpt = maximisation_step(cpt, bn)  
}
```

Si stima la **distribuzione dei missing data** cioè la distribuzione a posteriori dei loro possibili valori.

Si aggiornano i dati mancanti sulla base delle probabilità appena calcolate.

Si aggiornano le CPTs.

EM MANUALE (6)

STOP CRITERIA E EARLY STOPPING

- Vengono memorizzate le CPT dell'iterazione precedente e si fissa un parametro **alpha**.
- Per ogni probabilità condizionata, viene effettuata una differenza (in valore assoluto) tra i valori delle CPT precedenti con le CPT nuove. Questi valori vengono sommati tra di loro per determinare il **delta**.
- La procedura ritorna TRUE se **delta risulta essere minore di alpha**. Ritorna FALSE altrimenti.
 - Se la procedura ritorna TRUE l'algoritmo EM si arresta e produce in output le CPT finali;
 - se la procedura ritorna FALSE l'algoritmo EM continua con una nuova iterazione memorizzando le nuove CPT.
- Il limite massimo di iterazioni è dato un parametro fissato a priori e in questo caso è fissato a 1000.

Considerazioni

- La funzione `stopping_criteria` funziona su ogni tipi di struttura di rete;
- sull'esempio preso in considerazione, l'algoritmo si arresta alla quinta iterazione.

```
143 #works very well on every BN structures
144 stopping_criteria <- function(alpha) {
145   delta = 0
146   for (i in names(cpt)) {
147     #nodo prior
148     if (length(cpt[[i]]) == 2){
149       delta = delta + abs(as.numeric(cpt_last[[i]][1]) - as.numeric(cpt[[i]][1]))
150     }else{
151       cond = length(cpt[[i]])/2
152       num = 1
153       #determino numero di genitori di quel nodo. In futuro è migliorabile
154       while (cond != 2) {
155         cond = cond/2
156         num = num + 1
157       }
158     }
159     for (j in 1:num) {
160       delta = delta + abs(as.numeric(cpt_last[[i]][1,j]) - as.numeric(cpt[[i]][1,j]))
161     }
162   }
163 }
164
165
166 }
167 if (delta <= alpha){
168   return(TRUE)
169 }
170 else{
171   return(FALSE)
172 }
173 }
174
175 }
```

EM MANUALE (7)


In conclusione, l'algoritmo EM è strutturato come segue:

```
#Add alpha parameter (stop criterion)
cpt_last = cpt
for (i in 1:NUMBER_ITERATION) {
  M.prob = expectation_step()
  Data = update_data()
  cpt = maximisation_step(cpt, bn)
  STOP = stopping_criteria(0.001)
  if (STOP == TRUE){
    cat(sprintf("Uscita allo step: %s\n", i ))
    break
  }
  else{
    cpt_last = cpt
  }
}

print(cpt)
```


EM MANUALE (8)

I risultati ottenuti sono uguali a quelli pubblicati nelle slides «*Understanding Bayesian Networks*» di **Marco Scutari**.

Advanced Inference 

An Example: EM Algorithm, Fixed Structure (II)

1st Maximisation Step: we initialise the parameters of A and B using the complete observations.

$$\begin{array}{ll} \pi_{A,0} = 0.5 & \pi_{A,1} = 0.5 \\ \pi_{B,0|A,0} = 0.333 & \pi_{B,1|A,0} = 0.667 \\ \pi_{B,0|A,1} = 0.667 & \pi_{B,1|A,1} = 0.333 \end{array}$$


Note that this produces biased estimates if data are MNAR!

1st Expectation Step: we estimate the distributions of the missing data, that is, the (posterior) probabilities of their possible values (with `cpquery()` or `cpdist()` in `bnlearn`).

case	B	$\pi_{A,0 B}$	$\pi_{A,1 B}$
4	1	0.667	0.333
5	0	0.333	0.667
6	0	0.333	0.667

case	A	$\pi_{B,0 A}$	$\pi_{B,1 A}$
10	1	0.667	0.333

Marco Scutari University of Oxford

Advanced Inference 

An Example: EM Algorithm, Fixed Structure (III)


2nd Maximisation Step: we can then update the parameter estimates for A and B by summing up the observation indicators and the probabilities of the completions (say, π_{x_M}):

$$\pi = \frac{1}{n} \sum_{x_i} \mathbb{1}_O + \mathbb{1}_M \pi_{x_M}$$

The updated parameter estimates are:

$$\begin{array}{ll} \pi_{A,0} = 0.433 & \pi_{A,1} = 0.567 \\ \pi_{B,0|A,0} = 0.385 & \pi_{B,1|A,0} = 0.615 \\ \pi_{B,0|A,1} = 0.706 & \pi_{B,1|A,1} = 0.294. \end{array}$$

Marco Scutari University of Oxford

Advanced Inference 

An Example: EM Algorithm, Fixed Structure (IV)

2nd Expectation Step: using these updated parameter values, we can recompute the distributions of the missing values.

case	B	$\pi_{A,0 B}$	$\pi_{A,1 B}$
4	1	0.615	0.385
5	0	0.294	0.706
6	0	0.294	0.706

case	A	$\pi_{B,0 A}$	$\pi_{B,1 A}$
10	1	0.706	0.294

And so on, so forth ...

As the number of iterations increases, the **parameter updates gradually become smaller and smaller** until (after ≈ 4 iterations in this simple example) we can decide EM has converged and stop. We can set a threshold, for instance, by computing the **Kullback-Leibler distance** between the local distributions at two consecutive iterations.

Marco Scutari University of Oxford

EM MANUALE (9)

CONSIDERAZIONI FINALI

- Alla fine dell'esecuzione dell'algoritmo, i missing value vengono rimpiazzati con i seguenti valori:

	Data_1	Data_2	Data_3	Data_4	Data_5	Data_6	Data_7	Data_8	Data_9	Data_10
A	0	0	0	0	1	1	1	1	1	1
B	0	1	1	1	0	0	0	0	1	0

- Le CPT associate ai nodi risultano essere:

```
$A
A
    0      1
0.414713 0.585287

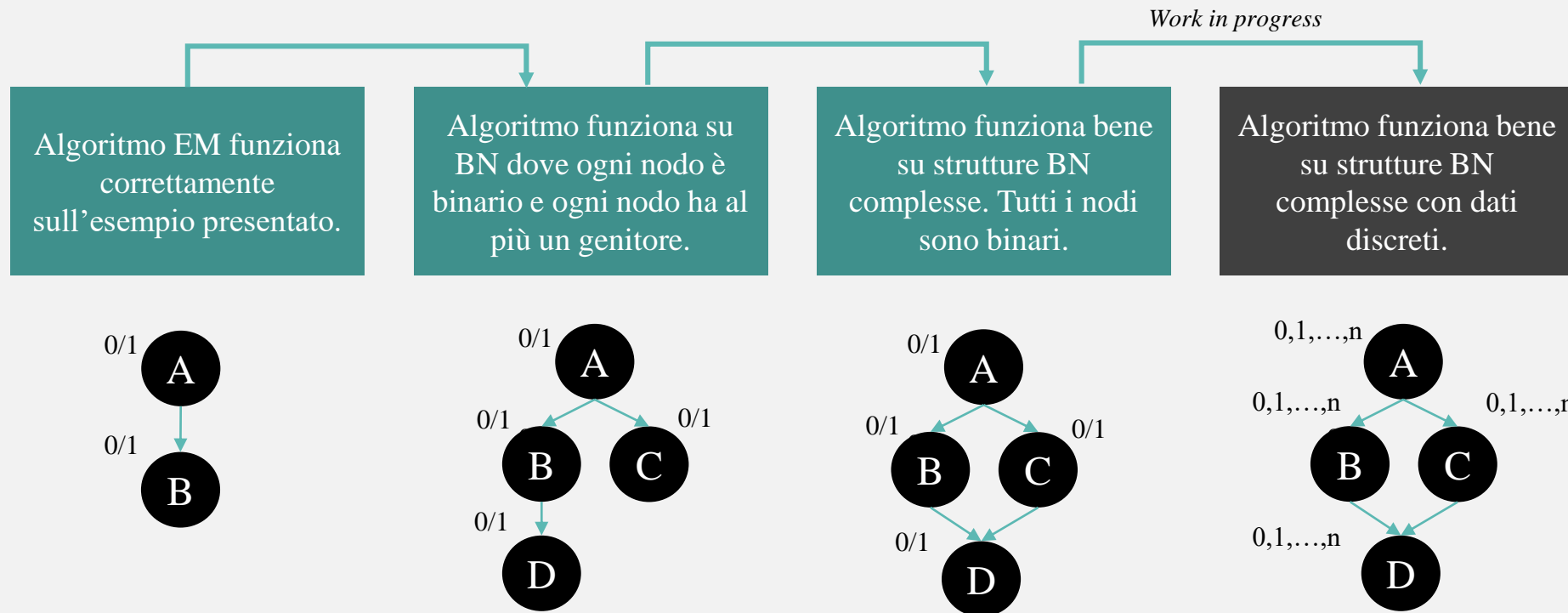
$B
A
    0      1
0 0.3710105 0.7132618
1 0.6289895 0.2867382
```

- L'algoritmo risulta convergere a questa soluzione già a partire dalla **quinta iterazione** (*stopping criteria*).
- Stima dei tempi computazionali:
 - **Expectation step:** $\theta(M)$ dove M è il numero di missing data presenti nel dataset.
 - **Updating step:** $\theta(i \cdot j)$ dove i è il numero di righe del dataset e j è il numero di colonne.
 - **Maximisation step:** $\theta(N \cdot V \cdot 2j)$ dove:
 - N è il numero di nodi della BN;
 - V è l'insieme dei possibili valori assumibili (in questo caso valori binari);
 - j è il numero di colonne del dataset;

EM MANUALE (10)

SVILUPPI FUTURI E PROSSIMI PASSI

1. Testare lo script R su diverse strutture di rete, dove ogni nodo è binario e ha al più un solo genitore.
2. Migliorare lo script in modo tale da computare l'algoritmo EM anche nel caso in cui un nodo ha più di un genitore (metodo di inferenza esatta).
3. Possibilità di operare con dati discreti e non solo binari.
4. Ottimizzazioni varie ed eventuali.



Pratica: R

Script EM con bnlearn

EM CON BNLEARN (1)

- Intero script realizzato in *R*.
- I passi di **Expectation** e di **Maximisation** sono stati eseguiti utilizzando le funzioni presenti in *bnlearn*.
- Il dataframe utilizzato è stato preso dalle slide «*Understanding Bayesian Networks*» di Marco Scutari.

Tre diversi approcci:

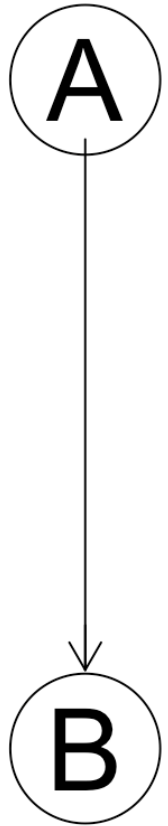
1. una BN con due nodi A e B ed un singolo arco $A \rightarrow B$; in questo caso, dove la struttura è già stata impostata, sono fornite le CPT presenti nelle slide e l'algoritmo EM converge in un'unica iterazione;
2. una BN con due nodi A e B ma senza nessun arco; in questo caso la struttura verrà imparata a partire dai dati; verrà utilizzato un *ciclo for* dove saranno chiamati i metodi di Expectation e di Maximisation;
3. verrà chiamata la funzione *structural.em* dandogli in input solo il dataframe.

N.B.: dove possibile sono stati mantenuti gli stessi parametri per tutti e tre i casi.

Ad esempio *maximize = "tabu"*, *whitelist = data.frame(from = "A", to = "B")*,
fit = "bayes", *method = "bayes-lw"*.

EM CON BNLEARN (2)

1. una BN con due nodi A e B ed un singolo arco $A \rightarrow B$



Bayesian network parameters

Parameters of node A (multinomial distribution)

Conditional probability table:

A	0	1
0.5	0.5	

Parameters of node B (multinomial distribution)

Conditional probability table:

A	0	1
B		
0	0.333	0.667
1	0.667	0.333

EM CON BNLEARN (3)

1. una BN con due nodi A e B ed un singolo arco $A \rightarrow B$

```
# expectation step
imputed = impute(bn, x, method = "bayes-lw")
```

```
# maximisation step (forcing A to be connected to B)
em.dag = tabu(imputed, whitelist = data.frame(from = "A", to = "B"))
bn = bn.fit(em.dag, imputed, method = "bayes")
bn
```

```
> imputed
  A B
1 0 0
2 0 1
3 0 1
4 0 1
5 1 0
6 1 0
7 1 0
8 1 0
9 1 1
10 1 0
> |
```

```
> bn
```

Bayesian network parameters

Parameters of node A (multinomial distribution)

Conditional probability table:

	0	1
	0.4090909	0.5909091

Parameters of node B (multinomial distribution)

Conditional probability table:

	A	
B	0	1
0	0.2777778	0.8076923
1	0.7222222	0.1923077

```
>
```

EM CON BNLEARN (4)

2. una BN con due nodi A e B ma senza nessun arco

```
> # initialise an empty BN
> imputed = x
> bn = bn.fit(empty.graph(names(x)), imputed)
> bn
```

Bayesian network parameters

Parameters of node A (multinomial distribution)

Conditional probability table:

	0	1
	0.4285714	0.5714286

Parameters of node B (multinomial distribution)

Conditional probability table:

	0	1
	0.5555556	0.4444444

```
> |
```

```
for (i in 1:4) {
  # expectation step
  imputed = impute(bn, x, method = "bayes-lw")
  imputed

  # maximisation step (forcing A to be connected to B,
  # and not to the other nodes because A create a self-loop)
  dag = tabu(imputed, whitelist = data.frame(from = "A", to = "B"))
  dag
  #graphviz.plot(dag)

  bn = bn.fit(dag, imputed, method = "bayes")
  bn
  # same results of first case
}
```

In questo caso non è possibile inizializzare i parametri della distribuzione locale di B poiché non è presente un arco da A a B.

EM CON BNLEARN (5)

2. una BN con due nodi A e B ma senza nessun arco

```
> imputed
```

	A	B
1	0	0
2	0	1
3	0	1
4	0	1
5	1	0
6	1	0
7	1	0
8	1	0
9	1	1
10	1	0

```
> |
```

```
> dag
```

Bayesian network learned via Score-based methods

model:

[A][B|A]

nodes:

2

arcs:

1

undirected arcs:

0

directed arcs:

1

average markov blanket size:

1.00

average neighbourhood size:

1.00

average branching factor:

0.50

learning algorithm:

Tabu Search

score:

BIC (disc.)

penalization coefficient:

1.151293

tests used in the learning procedure:

1

optimized:

TRUE

```
> |
```

EM CON BNLEARN (6)

2. una BN con due nodi A e B ma senza nessun arco

```
> bn
```

```
Bayesian network parameters
```

```
Parameters of node A (multinomial distribution)
```

```
Conditional probability table:
```

	0	1
A	0.4090909	0.5909091

```
Parameters of node B (multinomial distribution)
```

```
Conditional probability table:
```

	A	
B	0	1
0	0.2777778	0.8076923
1	0.7222222	0.1923077

```
> |
```



EM CON BNLEARN (7)

3. *funzione structural.em*

```
r = structural.em(x, fit = "bayes", fit.args = list(), maximize = "tabu",
  maximize.args = list(whitelist = data.frame(from = "A", to = "B")),
  "bayes-lw", impute.args = list(), return.all = TRUE,
  start = NULL, max.iter = 4, debug = FALSE)
```

\$imputed

	A	B
1	0	0
2	0	1
3	0	1
4	0	1
5	1	0
6	1	0
7	1	0
8	1	0
9	1	1
10	1	0

> |

\$dag

Bayesian network learned from Missing Data

model:

[A][B|A]

nodes: 2

arcs: 1

undirected arcs: 0

directed arcs: 1

average markov blanket size: 1.00

average neighbourhood size: 1.00

average branching factor: 0.50

learning algorithm: Structural EM

score-based method: Tabu Search

parameter learning method: Bayesian Parameter Estimation

imputation method: Posterior Expectation (Likelihood Weighting)

penalization coefficient: 1.151293

tests used in the learning procedure: 3

optimized: TRUE

EM CON BNLEARN (8)

3. *funzione structural.em*

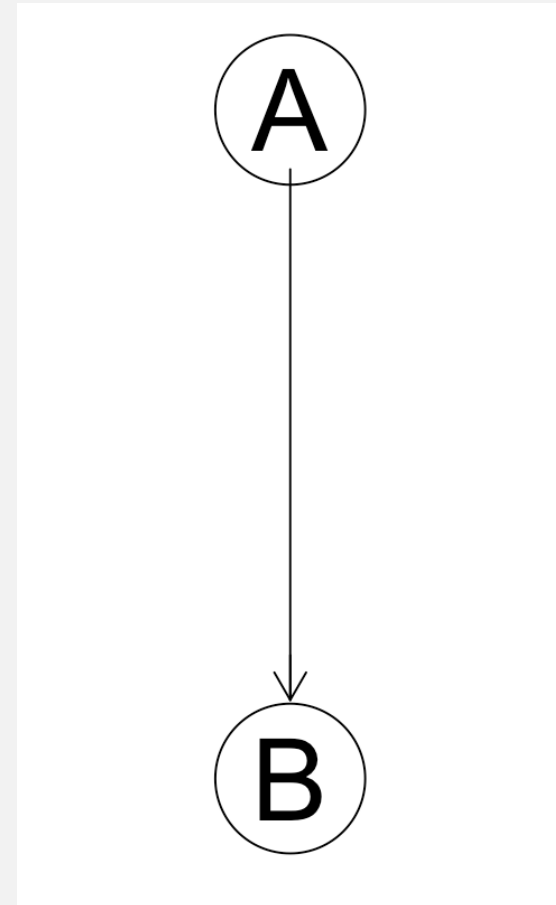
```
$fitted

Bayesian network parameters
Parameters of node A (multinomial distribution)
Conditional probability table:
      0      1
0.4090909 0.5909091

Parameters of node B (multinomial distribution)
Conditional probability table:

  A
B   0      1
0 0.2777778 0.8076923
1 0.7222222 0.1923077

> |
```



Pratica: R

Commento e confronto dei risultati

ANALISI DEI RISULTATI (1)

- Lo script **EM manuale** raggiunge i risultati attesi in base all'esempio preso in considerazione.
- Tuttavia, i risultati ottenuti dallo script **EM manuale** risultano essere diversi dallo script **EM con bnlearn**.

Risultati EM manuale

```
Bayesian network parameters
Parameters of node A (multinomial distribution)
Conditional probability table:
A
    0      1
0.4153445 0.5846555
Parameters of node B (multinomial distribution)
Conditional probability table:
  A
B   0      1
0 0.3720328 0.7126815
1 0.6279672 0.2873185
```

Risultati EM con bnlearn

```
$fitted
Bayesian network parameters
Parameters of node A (multinomial distribution)
Conditional probability table:
    0      1
0.4090909 0.5909091
Parameters of node B (multinomial distribution)
Conditional probability table:
  A
B   0      1
0 0.2777778 0.8076923
1 0.7222222 0.1923077
```

POSSIBILI MOTIVAZIONI

- Lo script **EM con bnlearn** potrebbe implementare una versione becera dell'algoritmo EM.
- Non sempre **EM con bnlearn** converge ad un'unica soluzione, se non si utilizza inferenza esatta.

ANALISI DEI RISULTATI

- Anche **EM con bnlearn**, nel metodo `imputed` raggiunge due risultati diversi sulla base della scelta degli iperparametri come il metodo di inferenza (**esatta** con `parents` oppure **approssimata** con `bayes-likelihood weighing`).
- Tramite **l'inferenza approssimata** è possibile raggiungere due risultati diversi:
 - lo stesso risultato ottenibile con il metodo `parents`;
 - un risultato significativamente diverso.

Risultati con `parents`

```
Bayesian network parameters
Parameters of node A (multinomial distribution)
Conditional probability table:
      0      1
0.3181818 0.6818182

Parameters of node B (multinomial distribution)
Conditional probability table:
  A      0      1
B 0 0.3571429 0.7000000
  1 0.6428571 0.3000000
```

Risultati con `bayes-lw`

```
Bayesian network parameters
Parameters of node A (multinomial distribution)
Conditional probability table:
      0      1
0.3181818 0.6818182

Parameters of node B (multinomial distribution)
Conditional probability table:
  A      0      1
B 0 0.3571429 0.7000000
  1 0.6428571 0.3000000
```

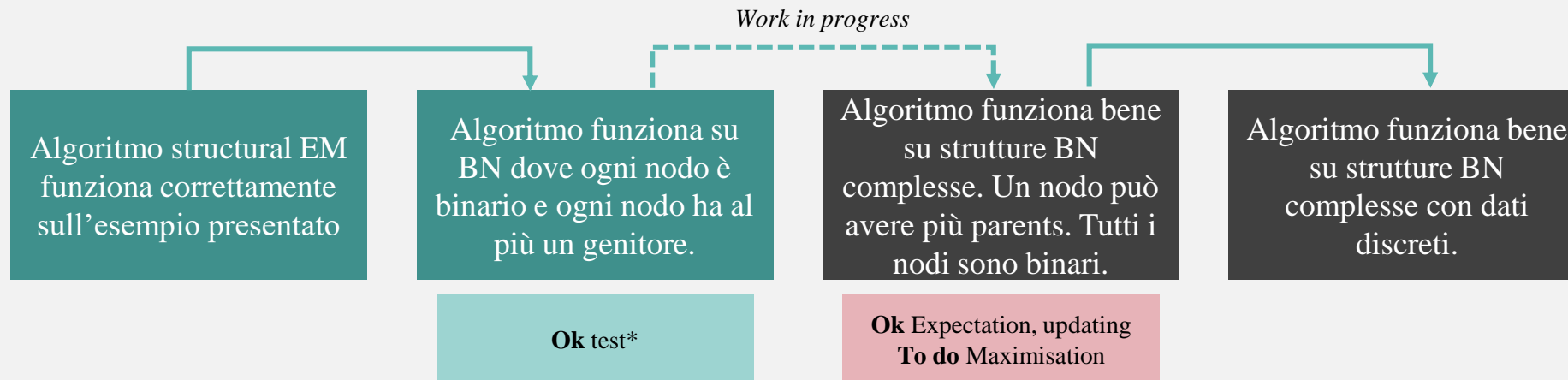
```
Bayesian network parameters
Parameters of node A (multinomial distribution)
Conditional probability table:
      0      1
0.4090909 0.5909091

Parameters of node B (multinomial distribution)
Conditional probability table:
  A      0      1
B 0 0.2777778 0.8076923
  1 0.7222222 0.1923077
```

EM MANUALE (10)

SVILUPPI FUTURI E PROSSIMI PASSI

1. Testare lo script R su diverse strutture di rete, dove ogni nodo è binario e ha al più un solo genitore.
2. Migliorare lo script in modo tale da computare l'algoritmo EM anche nel caso in cui un nodo ha più di un genitore (metodo di inferenza esatta).
3. Possibilità di operare con dati discreti e non solo binari.
4. Ottimizzazioni varie ed eventuali.



* Validazione manuale effettuando calcoli a mano.