UNIVERSITÀ DEGLI STUDI DI MILANO-BICOCCA

Dipartimento di Informatica Sistemistica e Comunicazione
Corso di Laurea Magistrale in Informatica
Scuola di Scienze

LEARNING OF BAYESIAN NETWORKS WITH MISSING DATA
Sintesi della tesi

Candidato
Ruggieri Andrea
Matr. 806808

Anno Accademico 2019–2020

Introduzione

La tesi si è focalizzata sull'implementazione e analisi dell'*algoritmo Expectation-Maximisation (EM*). Si tratta di un metodo likelihood-based avente lo scopo di rimpiazzare i valori mancanti presenti all'interno di un *dataset parziale*, ovvero contenente valori non osservati. Tale algoritmo si basa interamente sul concetto di *rete Bayesiana* ovvero un modello grafo-probabilistico che permette di rappresentare relazioni tra variabili. Una variante chiamata *Structural EM* viene talvolta impiegata per imparare la struttura della rete (ovvero le relazioni esistenti tra le variabili stesse). In questa tesi, tuttavia, l'attenzione si concentra sul rimpiazzo dei dati parziali e la struttura della rete Bayesiana sarà nota a priori.

Contesto esaminato

Il problema della presenza di dati mancanti è un problema serio che deve essere trattato con attenzione. Il motivo principale riguarda il fatto che la presenza di dati mancanti implica l'*assenza di informazione*. Questo potrebbe risultare un problema cruciale specialmente se queste informazioni risultano essere indispensabili, per esempio ai fini della diagnosi di una malattia, ai fini di condurre un'attenta analisi di bilancio, nel prendere decisioni... Rimpiazzare i valori mancanti è talvolta un'esigenza ed è importante applicare il miglior algoritmo che garantisce la miglior percentuale di rimpiazzamenti corretti. Nonostante in letteratura siano presenti diversi approcci per risolvere il problema dei dati mancanti, l'algoritmo EM si è dimostrato un'ottima alternativa. Per tale ragione è stato deciso di approfondire attentamente questo metodo.

Obbiettivi della tesi

Gli obbiettivi finali di questa tesi sono:

1. Valutare e sperimentare le diverse versioni dell'algoritmo EM variando la complessità del dataset e adottando diverse strategie per la generazione di valori mancanti;

2. Fornire al lettore indicazioni utili sulla base dei risultati che abbiamo osservato nella fase sperimentale, qualora esso abbia l'esigenza di rimpiazzare i valori mancanti;

3. Implementare un pacchetto R che renda disponibile il codice delle diverse versioni dell'algoritmo EM.

In particolare, le tre versioni dell'algoritmo EM che analizziamo all'interno della tesi prendono il nome di *Hard-Assignment EM*, *Soft-Assignment EM* e *Forced EM*. L'ultima versione risulta essere l'equivalente della versione *Soft-assignment*. Tuttavia, verrà rilassato il concetto che sta alla base del criterio di arresto, obbligando l'algoritmo ad arrestare

la sua esecuzione in un numero ridotto di iterazioni (numero paragonabile a quello della versione *Hard-Assignment*).

Stato dell'arte e importanza del progetto

In letteratura diversi approcci sono presenti. In primis, nel 1998, Friedman pubblicò: *the Bayesian Structural EM Algorithm* e questo articolo rimase un punto di riferimento in questo ambito. Tuttavia, tutti i risultati presenti in letteratura si soffermano esclusivamente alla versione *Hard* e mai nessun autore ha provato a implementare la versione *soft* in quanto ritenuto essere computazionalmente molto oneroso. Per tale motivo, in letteratura sono presenti esclusivamente risultati teorici della versione *Soft*. Il nostro scopo è quello di arricchire le conoscenze presenti in letteratura attraverso diverse esperimentazioni.

Approccio metodologico

Lo stage si è concentrato su tre momenti sequenziali: l'apprendimento di nozioni teoriche di base inerenti alle reti bayesiane, dati mancanti e all'algoritmo EM, l'implementazione del codice sorgente dell'algoritmo EM attraverso il linguaggio di programmazione R e la fase sperimentale dove è stato eseguito l'algoritmo EM con lo scopo di estrarre informazioni utili.

Algoritmo EM

L'algoritmo EM è una procedura iterativa. Ad ogni iterazione, vengono eseguiti due diversi step:

- *Expectation step (E-step)* si tratta di applicare l'algoritmo di inferenza esatta con lo scopo di stimare le probabilità a posteriori dei valori mancanti. Si tratta dello step computazionalmente più oneroso in quanto l'inferenza esatta è un algoritmo costoso.
- *Maximisation step (M-step)* vengono utilizzate le probabilità a posteriori calcolate nello step precedente con lo scopo di stimare i nuovi parametri della rete Bayesiana.

Listing 1: EM procedure using pseudocode

```
1 Begin Procedure EM_algorithm(...)
2    ∀θ ∈ BN set an initial value θ̂_0
3    while (|θ̂_{j-1} − θ_j| < ε or Nun_iterations) repeat:
4       E-Step: P(X_i^(M) | X_i^(O), θ̂_j) = P(X_i^(O)|X_i^(M),θ̂_j)P(X_i^(M),θ̂_j) / ∫ P(X_i^(O)|X_i^(M),θ̂_j)P(X_i^(M),θ̂_j)
5       M-Step: compute new θ̂_j given P(X_i^(M) | X_i^(O), θ̂_j)
```

6 $\theta = \hat{\theta}_j$
7 End Procedure

L'uso dell'inferenza esatta è fondamentale nell'algoritmo EM in quanto è alla base di una proprietà fondamentale:

> **Proprietà di convergenza.** L'algoritmo EM garantisce sempre la convergenza tuttavia:
>
> - Potrebbe convergere a un punto di massimo locale;
> - La convergenza potrebbe essere molto lenta;
> - Tutti gli step devono essere eseguiti attraverso l'inferenza esatta.

Per finire, la versione EM Hard si distingue dalla versione EM Soft sulla base dell'E-step. EM Soft considera tutti i possibili assegnmenti dei missing data. Al contrario, EM Hard seleziona l'assegnamento che risulta essere più probabile, diminuendo così i tempi computazionali e la quantità di memoria richiesta.

Implementazione

L'algoritmo EM è una procedura difficile da comprendere, inoltre è computazionalmente onerosa. Per questo motivo anche l'implementazione è un compito difficile che deve essere gestito e pianificato con estrema attenzione. L'implementazione di codice non ottimizzato potrebbe condurre a tempi di computazione eccessivamente alti, ciò si traduce in una difficile applicazione pratica. Lo sviluppo dell'algoritmo EM ha seguito un approccio bottom-up:

1. EM funziona correttamente su un esempio didattico molto semplice, composto da due soli nodi binari e un'unica relazione.
2. EM funziona correttamente su strutture molto semplici. I nodi sono tutti binari e ogni nodo ha al massimo un figlio e un solo genitore
3. EM funziona su strutture di reti Bayesiane più complesse. I nodi possono avere più figli e più genitori. Tuttavia, i nodi rimangono tutti binari.
4. EM funziona su tutti i tipi di reti Bayesiane discrete.

Inizialmente l'implementazione ha riguardato la versione EM Soft, successivamente è stata implementata la versione EM Hard.
Test di unità sono stati definiti ad ogni step con lo scopo di verificare la corretezza dell'algortimo implementato.

Fase sperimentale

La fase sperimentale ha permesso di raggiungere i primi due obbiettivi prefissati. La pianificazione di questa fase non è stata semplice in quanto ha coinvolto una serie di questioni la cui trattazione risulta essere fondamentale; tra le più importanti abbiamo:

- Quali dataset/reti devono essere scelti per adempiere a questa fase? Sono stati scelti dataset e reti che godono di proprietà diverse tra di loro. L'attenzione ad esempio si è focalizzata su reti di piccole/medie/grandi dimensioni, dataset fortemente etc. bilanciati/sbilanciati.

- Come sono stati generati i missing values partendo da dataset completi? E' stato necessario introdurre il concetto di **replica**.

- Su quali nodi i missing values devono essere generati? Questa domanda non è banale in quanto, con lo scopo di effettuare uno studio completo, non è sufficiente generare missing values su tutti i nodi selezionati causalmente, ma è necessario condurre analisi su certi tipi di nodi come ad esempio i nodi foglia, radici, più connessi etc.

- Quale meccanismo di dato mancante deve essere adottato? EM deve essere testato sia su dati MCAR sia MAR sia MNAR.

- Quando un dataset parziale può essere considerato valido per la nostra analisi? In questo caso è necessario fornire definizioni di validità.

- Come devono essere valutati gli esperimenti? Le metriche da noi presi in considerazione sono molteplici, ma la metrica di riferimento più importante è la Kullback-Leibler divergence.

$$KL[P(x)||P(y)] = \sum_{i}^{N} P(x_i) log \frac{p(x_i)}{p(y_i)}$$

Analisi dei risultati

L'analisi dei risultati ha permesso di estrarre informazioni utili. In particolare, esaminando attentamente tutti i risultati, è stato possibile ricavare un albero di decisione che ha il compito di aiutare il lettore a scegliere la versione dell'algoritmo EM più corretto sulla base del contesto in cui esso si trova. Tale albero è mostrato nella figura 0.1. Per ogni caso osservato, si consiglia al lettore una determinata versione dell'algoritmo EM e un certo numero di iterazioni. Ad esempio per reti di piccole dimensioni, non è necessario fissare un numero troppo alto di iterazioni ma in alcuni contesti le due versioni dell'algoritmo EM potrebbero esibire risultati molto diversi.

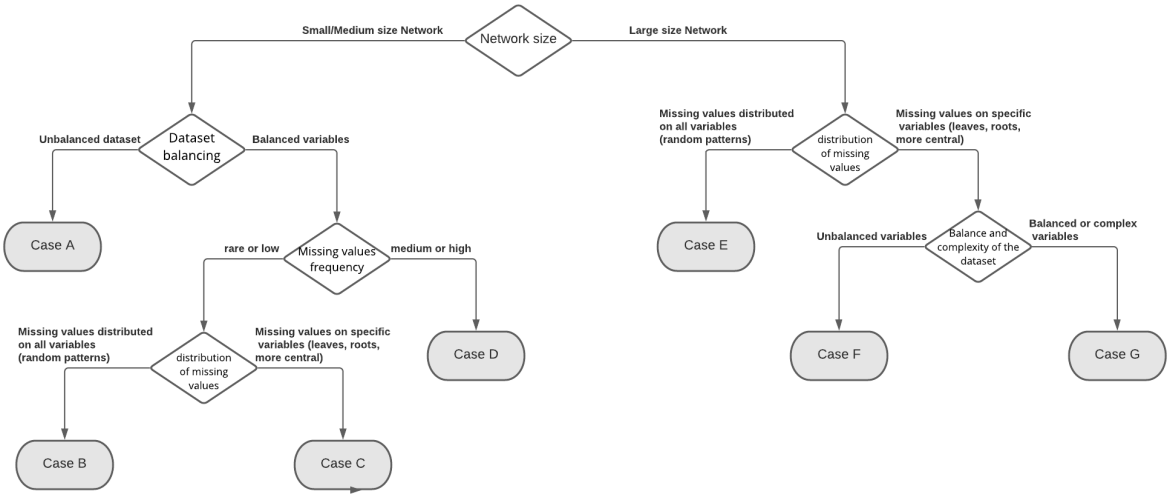

Figura 0.1: Albero decisionale ottenuto dallo studio dei risultati

Considerazioni finali e ringraziamenti

Il lavoro di stage è stato svolto con il team MADLAB dell'università Bicocca di Milano. L'intero lavoro è stato seguito con attenzione dall'IDSIA, e in particolare, con il Ph.D. Marco Scutari, fondatore e manutentore della liberia `bnlearn`. A tal proposito si ringraziano tutti i membri: il Prof. *Fabio Stella* per la sua grande disponibilità e il suo grande impegno. Ci ha fatti crescere e amare il mondo del data mining, dapprima con il suo corso magistrale e successivamente con questo lavoro di stage. Il Ph.D. *Marco Scutari* per i suoi preziosi consigli e la sua disponibilità nel seguire le video-presentazioni ogni tre settimane. Il mio stretto collega *Francesco Stranieri* che mi ha aiutato e seguito in tutto il percorso di stage e condiviso alcuni progetti universitari. Il mio co-relatore *Alessandro Bregoli* e tutti gli altri membri del progetto: *Alessio* ed *Emanuele*.

L'intero progetto è dispobile su GitHub in forma open-source (accessibile a tutti). All'interno della repository di riferimento [1] è possibile trovare:

- L'implementazioni delle versioni dell'algoritmo EM;
- I più significativi test di unità che ci hanno permesso di valutare la correttezza dei codici sorgente;
- I risultati ottenuti dall'esecuzione di ciascun esperimento.

[1] GitHub repository: https://github.com/madlabunimib/Expectation-Maximisation