PROGETTO SU INCERTEZZA

IALab A.A. 19/20

Amedeo Racanati 928995

Angelo Pio Sansonetti 928869

Introduzione

Lo scopo del progetto è stato quello di implementare algoritmi di variable elimination, basandosi su differenti metodi di ordinamento dei nodi. Sono stati implementati algoritmi di inferenza esatta di reti Bayesiane sia statiche che dinamiche.

Per le reti statiche sono state adottate delle tecniche di pruning, con le quali è stato possibile in alcuni casi velocizzare il processo di inferenza.

Nel seguito saranno esposte le performance ottenute dai differenti tipi di algoritmi per differenti tipologie di reti e saranno accompagnate da brevi commenti.

Metodi di ordinamento

Qui di seguito verranno trattati tre tipi di ordinamento utilizzati per l'algoritmo di *variable elimination*.

Il primo tipo di ordinamento è il topologico inverso, che semplicemente ordina le variabili partendo dalle foglie salendo fin su fino alle radici. L'implementazione in questo caso ci è stata già fornita dalla libreria aima-core.

Il secondo tipo di ordinamento è il min-degree, che ordina i nodi secondo la loro connettività. In questo ordinamento come nel successivo, un nodo si considera connesso ad un altro nodo se compare almeno in uno dei fattori. Per definizione quindi un nodo si considera connesso ai nodi del proprio Markov Blanket, ossia i propri genitori, i propri figli e i genitori dei propri figli.

Nel programma si utilizza una struttura dati ad hoc, che consiste in una lista di nodi e per ciascuno di essi si indica l'insieme dei nodi vicini (classe *SimpleNode*). Partendo dalla rete iniziale, si provvede a trovare il nodo che possiede meno connessioni, dopodiché lo si elimina dalla rete e si ricalcola la connettività dei nodi nella nuova rete in maniera iterativa, finché sono stati restituiti tutti i nodi in ordine.

Nell'ordinamento min-fill si ordinano i nodi in maniera tale da mantenere il più basso grado possibile di connettività della rete, con la conseguenza di gestire dei fattori più piccoli. In questo caso si trova il nodo che una volta rimosso dalla rete, genera il minor numero di nuovi archi possibile. Preso in esame un nodo (X0), il numero di nuovi archi creati viene calcolato considerando tutti i nodi adiacenti al nodo considerato, e per ciascuno (X1) di essi si vede se X1 è già connesso con i nodi adiacenti a X0. Se X1 è già connesso ai nodi adiacenti di X0 allora non saranno creati nuovi archi, altrimenti sì. Una volta ottenuto il nodo ottimale, lo si rimuove dalla rete e si provvede a ricalcolare lo score dei nodi rimasti, basandosi sulla nuova rete creata.

Infine c'è da dire che l'algoritmo di variable elimination, date le differenti modalità di ordinamento, provvede ad effettuare una sum-out di una variabile quando è certo che la variabile in questione è comparsa in tutti i fattori ad essa relativi (le variabili di query non vengono considerate per la sum-out).

I criteri di ordinamento sopra citati sono stati utilizzati sia per le reti statiche che per quelle dinamiche.

Pruning

La prima tecnica di pruning utilizzata è quella che ha rimosso dalla rete tutti quei nodi che non sono antenati di un nodo di evidenza o di query. L'unica cosa da dire è che per ottimizzare l'algoritmo si è partiti in ordine topologico inverso, includendo soltanto quei nodi di evidenza o di query, o loro antenati.

La seconda tecnica di pruning è quella relativa alla costruzione del moral graph. L'algoritmo parte da ciascun nodo (X0) di query e fa una ricerca considerando i nodi del Markov Blanket del nodo X0, i quali vengono a loro volta espansi qualora non siano nodi di evidenza. In questa maniera si ottengono tutti quei nodi che sono collegati ai nodi di query espandendo le varie Markov Blanket, rimuovendo tutti i nodi non raggiunti.

Per quanto riguarda la rimozione degli archi, si è provveduto a creare dei nodi che avessero come genitori soltanto quei nodi che non fossero di evidenza. In questo modo si riduce la CPT dei nodi che hanno qualche nodo di evidenza come genitore iniziale.

Il pruning è stato adottato per le sole reti statiche, in quanto per quelle dinamiche tutti i nodi foglia rappresentano variabili di evidenza, mentre i nodi radice sono variabili di stato. Pertanto le tecniche di pruning sono inutili.

Rollup filtering

L'algoritmo di rollup filtering è stato implementato per l'inferenza esatta nelle reti bayesiane dinamiche. Fissati N passi temporali e date le evidenze per ciascuno di questi passi, quel che vien fatto è di calcolare i fattori di tutte le variabili di stato, raggruppate opportunamente.

Innanzitutto vengono calcolati i fattori per ogni singola variabile del passo 0, e viene determinato l'ordinamento delle variabili durante l'eliminazione delle stesse, che è uguale per ciascun passo. Per ogni passo si parte quindi con dei fattori già presenti per le variabili del passo precedente, si provvede ad effettuare la variable elimination delle variabili dello stato precedente, di quelle hidden e di quelle di evidenza dello stato corrente, evitando di effettuare la sum-out delle variabili dello stato corrente.

Una volta fatto ciò si raggruppano tra loro quei nodi che compaiono almeno in uno stesso fattore. Dopodiché vengono restituiti i valori dei relativi fattori che rappresentano le probabilità delle variabili dello stato corrente; infine si passa allo stato successivo considerando questi fattori come facenti riferimento alle variabili dello stato precedente.

L'algoritmo implementato è stato confrontato con l'algoritmo (particle filtering) già presente nella libreria aima-core, al fine di valutarne la correttezza. Si è notato che se si vuole avere una buona approssimazione col particle filtering, questo algoritmo è più lento rispetto al nostro.

Reti bayesiane

Per i nostri esperimenti abbiamo definito 13 reti statiche e 7 reti dinamiche.

Ciò che varia in queste reti è il numero dei nodi così come il numero della complessità computazionale.

Mentre per le reti dinamiche le variabili di query e di evidenza sono fisse (le prime corrispondono alle variabili di stato mentre le seconde ai nodi foglia), nelle reti statiche si sono potute sperimentare diverse configurazioni per quanto riguarda i nodi di query e di evidenza.

In particolare sono stati definiti tre scenari:

1. Nodi di query vicini alla radice mentre nodi di evidenza vicini alle foglie (denominato firstQuery);
2. Nodi di query vicini alle foglie mentre nodi di evidenza vicini alla radice (lastQuery);
3. Nodi di query presenti a livello intermedio della rete e nodi di evidenza inseriti vicino alla radice o alle foglie (mediumQuery).

La determinazione di quali nodi dovessero essere di query o di evidenza è stata fatta in maniera automatica. Anche l'assegnazione dei valori delle evidenze è stata fatta in maniera casuale, prendendo il dominio del nodo selezionato e scegliendo un valore a caso su quelli disponibili.

Risultati – Reti Bayesiane statiche

I dati dei risultati per le reti statiche sono presenti nel file "risultati_statiche_tempo.csv". I tempi raccolti sono espressi in millisecondi.

La differenza tra i diversi metodi di ordinamento emerge quando le reti sono di dimensioni medio-grandi. Quando la rete è medio-piccola, non sempre è conveniente effettuare un ordinamento differente dal topologico inverso, in quanto l'overhead per calcolare un differente tipo di ordinamento non corrisponde ad un guadagno dei tempi. Tendenzialmente il min-degree ha delle perfomance migliori rispetto al min-fill, e queste due sono nettamente migliori della topologica inversa quando la rete diventa grande.

Per quanto riguarda il pruning, si è osservato che se viene effettuato su reti di dimensione piccola ha comportato uno svantaggio in termini di tempo per tutti e tre i metodi di eliminazione. Quando le dimensioni della rete sono più grandi, il pruning giova in maniera più netta per l'ordinamento topologico inverso, mentre per i restanti tipi di ordinamento il vantaggio è più piccolo, sebbene presente.

Osservando i diversi tipi di scenari, si osserva come nella maggior pare delle volte i tempi di elaborazione aumentano in maniera evidente quando si adopera il mediumQuery. Per le grosse reti firstQuery comporta generalmente dei tempi di elaborazione minori, anche se ciò non è sempre vero.

Infine confrontando le diverse reti, si nota che i tempi di elaborazione aumentano all'aumentare del fattore più grande della rete (confrontare *WIN95PTS* ed *hepar2*), mentre il numero dei nodi incide in maniera meno influente.

Risultati – Reti Bayesiane dinamiche

Rete	Nodi	Fattore Massimo	Reverse EliminationAsk (ms)	MinDegree EliminationAsk (ms)	MinFill EliminationAsk (ms)
Alarm	49	5	12.521	9.321	14.273
Asia	10	3	9	6	7
Child	24	3	510	395	240
Earthquake	7	3	2	2	2
Insurance	29	4	29.514	1.669	23.689
Sachs	13	4	7	7	8
Water	38	7	22.906	59.372	47.342

Anche per le reti dinamiche, si può notare come il fattore massimo incide sui tempi di elaborazione (confrontare *child* e *insurance*) mentre il numero dei nodi per ogni step in queste reti ha influenza.

Quando i nodi sono interconessi tra loro in maniera uniforme, avendo ciascuno di essi un numero di archi similare, gli ordinamenti *minFill* e *minDegree* non trovano un ordinamento dei nodi migliore, comportando un overhead e quindi dei tempi di elaborazione più lunghi (si veda *water*).

L'ordinamento min-degree ha tendenzialmente degli score migliori, mentre il min-fill in questo caso è peggiore dell'ordinamento topologico inverso.