Inferenza su reti Bayesiane con Hugin Propagation

Relazione elaborato Intelligenza Artificiale - a.a. 2019/2020

Diego Biagini

August 30, 2020

1 Introduzione

Una rete Bayesiana è un tipo di modello grafico utile per rappresentare l'incertezza intrinseca di un sistema reale costituito da vari componenti, insieme ai legami tra di essi.
Il loro uso principale è quello di sfruttare dipendenze e indipendenze tra variabili aleatorie per sapere come la certezza di un evento influenza la probabilità di un altro.
Uno dei modi per trattarle è quello di registrare l'evento conosciuto e propagare questa conoscenza attraverso un grafo relativo alla rete.
Lo scopo di questo progetto è quello di realizzare un particolare tipo di operazione di propagazione, detta Hugin propagation, come descritta da F.Jensen.[1]

2 Hugin propagation

Per prima cosa sono state implementate le varie strutture dati necessarie alla rappresentazione di una rete Bayesiana, ovvero le variabili aleatorie(che rappresenteranno anche i nodi della rete), le tabelle di probabilità condizionata e la rete vera e propria, vista come un grafo diretto.
Successivamente è stata realizzata la struttura del *JunctionTree* e dentro di esso sono stati inseriti i meccanismi che consentono l'inserimento dell'evidenza e la sua propagazione.

2.1 Componenti di base

2.1.1 Variabili aleatorie

E' stato scelto di identificare univocamente una variabile aleatoria con un nome e con un insieme di valori, i valori che essa può assumere. La classe che la rappresenta è *Variable*.

2.1.2 Belief Table

Le tabelle di probabilità condizionata, relative ad un insieme di variabili aleatorie.
Nell'implementazione non è stata definita la relazione precisa tra le variabili della tabella(una tabella sulle variabili A e B può definire sia $\mathbb{P}(A|B)$ sia $\mathbb{P}(B|A)$).
Per poterle usare nella propagazione sono state definite le operazioni:

- Moltiplicazione tra tabelle: date due tabelle t_{AB} e t_{BC} è possibile moltiplicarle e il risultato è una tabella sulle variabili ABC tale che $t_{AB} \cdot t_{BC}(a,b,c) = t_{AB}(a,b) \cdot t_{BC}(b,c)$
- Divisione tra tabelle: analogo alla moltiplicazione, stando attenti alle divisioni $\frac{0}{0}$, il cui risultato deve essere imposto a 0.
- Marginalizzazione di una tabella su un insieme di variabili: se t_V è una tabella e $W \subseteq V$ è possibile marginalizzarla su W secondo la formula $t_W = \sum_{V \setminus W} t_V$.

[1] F.Jensen. *Introduction to Bayesian Networks*. 1997.

2.1.3 Bayesian Net

Memorizzata come grafo diretto, attraverso un dizionario, i cui nodi sono *Variable*s.
Dopo aver aggiunto variabili e dipendenze tra di esse è necessario aggiungere le *BeliefTable* relative ad ogni nodo, controllando che la tabella assegnata ad un nodo sia definita sulla variabile del nodo e sulle variabili dei nodi padre.

2.2 Da rete Bayesiana a Junction Tree

La Hugin propagation non lavora sulla rete bayesiana, ma su una struttura duale detta junction tree, i cui nodi sono insiemi di variabili aleatorie(cricche), a differenza della rete bayesiana.
Anche gli archi sono etichettati da un insieme di variabili(le variabili condivise dai nodi collegati) e sono detti separatori.
Per tradurre il primo modello nel secondo sono necessari i seguenti passi:

- Costruzione del junction tree a partire dalla rete bayesiana(non trattato in questo progetto) e inizializzazione delle tabelle di ogni cricca e separatore(tutte le probabilità sono impostate ad 1)
- Scelta di una cricca per ogni variabile, tale cricca deve contenere la variabile e tutti i suoi padri
- Assegnazione della *BeliefTable* di ogni variabile alla cricca scelta

Usando una struttura di questo tipo è necessario decidere come consultare le probabilità che una variabile assuma ciascuno dei suoi valori, per fare questo basta estrarre la *BeliefTable* della cricca scelta per tale variabile durante la fase di creazione e marginalizzare tale tabella sulla variabile.
Perchè questo metodo funzioni fin da subito è necessario svolgere un round iniziale di propagazione dell'evidenza, eseguito simulando l'inserimento di evidenza in ogni variabile e la successiva Hugin propagation.

2.3 Inferenza su Junction Tree

2.3.1 Inserimento dell'evidenza

Inserire evidenza nel *JunctionTree* che una certa variabile **A** può solo assumere il valore A_i significa impostare le *BeliefTable* relative ad **A** in modo che gli stati diversi da A_i siano impossibili(probabilità 0). Allo stesso modo se vogliamo sapere le probabilità dell'universo(che supponiamo sia nota) data l'evidenza possiamo usare la legge di Bayes per ottenere:

$$\mathbb{P}(U|e) = \frac{\mathbb{P}(U,e)}{\mathbb{P}(e)} = \frac{\mathbb{P}(U,e)}{\sum_U \mathbb{P}(U,e)}$$

Però così è necessario conoscere $\mathbb{P}(U)$, il che comporta la gestione di tabelle troppo grandi per un grande numero di variabili. Questo problema viene risolto nella Hugin Propagation mantenendo le tabelle separate nel *JunctionTree*, inserire l'evidenza dentro di esse e successivamente propagarla alle altre tabelle.

2.3.2 Propagazione dell'evidenza

La propagazione si basa sul passaggio di informazioni tra nodi del junction tree, ottenuto attraverso l'operazione di assorbimento: date due cricche **V** e **W**,separate dal separatore **S**, **W** assorbe informazioni da **V** con le seguenti operazioni:

- $t_s^* = \sum_{V \setminus W} t_v$
- $t_w^{new} = t_w^{old} \cdot \frac{t_s^*}{t_s}$
- $t_s = t_s^*$

La Hugin Propagation definisce a questo punto uno schema di passaggio dei messaggi che ottimizza il numero di assorbimenti che devono essere eseguiti. Questo schema fa uso di due funzioni *CollectEvidence* e *DistributeEvidence*, che si occupano rispettivamente di portare l'informazione dalle foglie dell'albero a una radice e viceversa.
Queste operazioni sono state implementate attraversando il junction tree con un attraversamento in ampiezza, applicando assorbimenti tra nodi durante esso.
In conclusione i passaggi eseguiti per compiere un round di scambio messaggi e quindi aggiornare tutte le tabelle sono:

- Scegliere una qualsiasi cricca **R** come radice
- Eseguire le operazioni di *CollectEvidence* e *DistributeEvidence* scegliendo come radice **R**
- Normalizzare le tabelle del junction tree, dividendo ogni cella per $\sum_A \mathbf{P}(A)$, per qualsiasi variabile A

3 Esperimenti

Gli esperimenti sono stati eseguiti su diversi modelli trovati nella cartella *Samples* del programma Hugin Expert.

3.1 Confronto con Hugin Expert

Analizziamo la fedeltà dell'implementazione dell'algoritmo confrontando i risultati con quelli del programma Hugin Expert.
E' possibile ricostruire il modello voluto usando le varie funzioni contenute nel file *models.py*, oppure caricando i modelli dai file presenti nella cartella models, tramite la funzione *util.load_model*

3.1.1 Monty Hall

Modello che rappresenta il classico "Monty Hall problem".
Le tre variabili della rete possono assumere tre valori("door1", "door2", "door3") ed indicano rispettivamente la porta scelta dal concorrente(**F**),la porta aperta da Monty(**M**) e la porta dietro cui si trova il premio(**P**).

	Risultato	Hugin Expert
$\mathbb{P}(P\|F = door1, M = door2)$	$door1 = 0.3333, door2 = 0.00, door3 = 0.6667$	$door1 = 0.3333, door2 = 0.00, door3 = 0.6667$
$\mathbb{P}(P\|F = door3, M = door1)$	$door1 = 0.00, door2 = 0.6667, door3 = 0.3333$	$door1 = 0.00, door2 = 0.6667, door3 = 0.3333$
$\mathbb{P}(M\|P = door2, F = door1)$	$door1 = 0.00, door2 = 0.00, door3 = 1.00$	$door1 = 0.00, door2 = 0.00, door3 = 1.00$

In questo modello si può notare come sia possibile inserire evidenza contrastante(per esempio $F = door1, M = door1$, Monty non può aprire la stessa porta aperta dal concorrente) ,casistica che deve essere presa in considerazione e considerata come un inserimento errato.

3.1.2 Chest Clinic

Modello visto a lezione, riguardante una clinicache si occupa di problemi respiratori.
Le variabili della rete ci dicono: se il paziente è stato in Asia(**A**), se ha tubercolosi(**T**), se è fumatore(**S**),se ha bronchite(**B**),se ha un cancro ai polmoni(**L**), se soffre di dispnea(**D**),se ha un cancro oppure tubercolosi(**E**), se il risultato dei raggi x è positivo(**X**).
Ogni variabile può solo assumere i valori "yes" o "no".

	Risultato	Hugin Expert
$\mathbb{P}(T\|A = yes, X = yes, D = no)$	$yes = 0.2224, no = 0.7776$	$yes = 0.2224, no = 0.7776$
$\mathbb{P}(T\|A = no, X = yes, D = no)$	$yes = 0.0520, no = 0.9480$	$yes = 0.0520, no = 0.9480$
$\mathbb{P}(L\|S = yes, X = yes, D = yes)$	$yes = 0.7237, no = 0.2863$	$yes = 0.7237, no = 0.2863$

3.1.3 Stud Farm

Modello che rappresenta una fattoria contenente 12 cavalli(le variabili della rete, identificate dalle lettere da **A** a **L**), che possono essere portatori oppure no di un gene recessivo che causa una malattia.
Il cavallo **J** può essere "carrier","pure" o "sick"(indicati come c,p,s nella tabella); gli altri cavalli possono essere "carrier" o "pure".

	Risultato	Hugin Expert
$\mathbb{P}(J\|H = p, I = c)$	$s = 0.0000, c = 0.5000, p = 0.5000$	$s = 0.0000, c = 0.5000, p = 0.5000$
$\mathbb{P}(J\|A = c, C = c, E = p, K = p)$	$s = 0.0262, c = 0.2818, p = 0.6920$	$s = 0.0261, c = 0.2816, p = 0.6923$
$\mathbb{P}(J\|A = c, B = c, C = c, K = c, L = c)$	$s = 0.0672, c = 0.3841, p = 0.5487$	$s = 0.0672, c = 0.3841, p = 0.5487$

3.2 Velocità di inferenza

Una metrica di interesse è anche l'efficienza della Hugin Propagation, in particolare il guadagno che essa porta in termini di velocità di esecuzione rispetto ad un approccio naive. Prendiamo quindi in considerazione anche un modo di calcolare le probabilità delle variabili semplicemente moltiplicando tutte le tabelle della rete tra esse e marginalizzando sulla variabile di interesse.
Analizziamo le prestazioni dei due metodi su tre modelli di complessità crescente, inserendo evidenza e richiedendo le probabilità di una qualsiasi variabile(in entrambi i casi la variabile richiesta non è importante).
E' possibile riprodurre i risultati caricando il modello voluto nel file *profiler.py*.

Modello	Inserimenti di evidenza	Hugin propagation	Prodotto tra tabelle
fire	R=true	0.002s	0.011s
	R=true, S=false	0.002s	0.011s
	R=true, S=false, A=true	0.002s	0.012s
studfarm	J=sick	0.004s	5.407s
	J=sick, A=carrier	0.007s	5.281s
	J=sick, A=carrier, E=pure	0.009s	5.227s
poker	FC=1 changed	0.007s	21.844s
	FC=1 changed, SC=0 changed	0.012s	21.245s
	FC=1 changed, SC=0 changed, MH=flush	0.024s	21.742s

4 Conclusioni

E' stato constatato il corretto funzionamento del programma, il quale restituisce sempre dati concordi col programma Hugin Expert.
Inoltre è stato constatato come l'uso di tale propagazione dell'evidenza sia estremamente più veloce rispetto a un semplice prodotto tra tabelle.