Entailment in logica proposizionale

Giulia Paone

In questo progetto si implementano gli algoritmi di backward e forward chaining e si testano nell'immaginario "Wumpus world".

1 Introduzione

Il concetto di entailment, o implicazione logica, definisce, in logica, un legame tra sentenze. Si dice che una sentenza è implicata logicamente da un'altra o da un gruppo di altre quando è vera in ogni modello in cui queste sono vere, intendendo per modello una rappresentazione più o meno astratta della realtà. Il processo con cui viene verificato l'entailment è detto inferenza.

2 Forward e backward chaining

Un concetto alla base della logica proposizionale è quello di clausola: una clausola è una disgiunzione logica fra letterali, intendendo con letterale una formula atomica o la sua negazione. Un tipo particolare di clausola è la clausola di Horn in cui al più un letterale è positivo. Queste possono essere utilizzare negli algoritmi forward e backward chaining per ottenere inferenza logica, tramite un'applicazione ripetuta della regola "modus ponens". Infatti, le clausole di Horn possono essere scritte equivalentemente come implicazioni logiche, in cui i letterali negati costituiscono le premesse. Partendo da una base di conoscenza costituita da letterali positivi e da una serie di implicazioni logiche di cui i primi sono le premesse, si può applicare il modus ponens.

2.1 Caratteristiche e differenze

Nel progetto vengono implementati i due algoritmi per poterne osservare le differenze. L'algoritmo di forward chaining è del tipo data-driven ed esegue una ricerca breadth-first, ovvero parte dai fatti conosciuti ed applica iterativamente le regole di inferenza per ottenere nuovi dati, fino ad arrivare alla query. In questo modo viene esplorato ogni ramo di inferenza, anche quelli non necessari al nostro fine. Al contrario, l'algoritmo di backward chaining è del tipo goal-driven ed esegue una ricerca depth-first, ovvero scende dal goal verso i fatti conosciuti, attraversando solo i rami necessari a trovare premesse vere per poter dimostrare il goal.

3 Il progetto

3.1 Applicazione al wumpus world

Il wumpus world è un mondo immaginario modellato su una griglia 4X4, in cui l'agente protagonista deve trovare la casella contenente l'oro, evitando il terribile "wumpus" e i tranelli nascosti nel percorso. L'agente può riconoscere i pericoli che lo circondano grazie ai dati raccolti tramite sensori.

Nel progetto vengono proposti esempi di inferenza a partire da una base di conoscenza costituita dalle regole del wumpus world ed alcune percezioni.

Di seguito vengono elencati i fatti, le possibili percezioni e le regole del wumpus world.

	Simbolo	Significato
Fatti	P_{ij}	Pit (fossa) nella casella ij
	W_{ij}	Wumpus nella casella ij
	V_{ij}	La casella ij è stata visitata
	G_{ij}	La casella ij contiene l'oro
	OK_{ij}	La casella ij è sicura
Percezioni	B_{ij}	Nella casella ij viene percepita brezza
	S_{ij}	Nella casella ij viene percepito stench (puzza)
	Gl_{ij}	Nella casella ij viene percepita polvere dorata
Regole	$B_{ij} \Rightarrow P_{(i+1)j} \vee P_{(i-1)j} \vee P_{i(j+1)} \vee P_{i(j-1)}$	Se viene percepita brezza nella cella ij, ci sarà una fossa in una delle celle limitrofe, raggiungibili orizzontalmente o verticalmente
	$S_{ij} \Rightarrow W_{(i+1)j} \vee W_{(i-1)j} \vee W_{i(j+1)} \vee W_{i(j-1)}$	Se viene percepita puzza nella cella ij, ci sarà il wumpus in una delle celle limitrofe, raggiungibili orizzontalmente o verticalmente
	$Gl_{ij} \Rightarrow G_{ij}$	Se viene percepita polvere dorata nella cella ij, ci sarà l'oro nella cella ij

3.2 Costrutti utilizzati

Per poter analizzare i due algoritmi di inferenza sono stati definiti dei costrutti all'interno del programma, al fine di rappresentare i letterali, le sentenze e la base di conoscenza.

- Literal: ogni letterale è istanza della classe Literal. Essa ne descrive il nome e, attraverso il nome stesso, se si trova in forma negata o meno.
- Sentence: ogni sentenza è istanza della classe Sentence. Si sottintende che le sentenze siano scritte nella forma di clausola di Horn o, equivalentemente, di implicazione logica. Infatti, la classe contiene due liste di letterali, la prima rappresentante le premesse e la seconda le conclusioni. In più, vi è un attributo che specifica il nome della regola ed uno che indica il numero di letterali presenti nella premessa.

- KnowledgeBase: la base di conoscenza è istanza della classe KnowledgeBase. Questa contiene una lista di simboli ed una di regole.

Nel contesto del wumpus world è stata utilizzata la seguente sintassi: ogni percezione, tra le possibili, è indicata da una lettera e due numeri che indicano la posizione della cella sulla griglia. In modo analogo sono rappresentati i fatti conosciuti. In più, viene aggiunta la lettera 'n' davanti al nome per inidicare la negazione. Ad esempio l'oggetto **nB11** è istanza di un letterale che rappresenta l'assenza di brezza nella casella (1,1).

3.3 Implementazione degli algoritmi

3.3.1 Forward Chaining Entailment

L'algoritmo di forward chaining viene qui implementato sotto il nome **FCEntails**. Questa funzione prende come argomenti la base di conoscenza e la query che vogliamo inferire e restituisce un valore booleano uguale a 'True' solo se la query in questione viene inferita. Due dizionari, chiamati 'count' e 'inferred', tengono traccia rispettivamente del numero di premesse presenti in ogni regola e dei letterali inferiti. Una coda LIFO, chiamata 'queue', contiene i letterali che sappiamo già essere veri, cioè i fatti conosciuti.

Il processo di inferenza è contenuto in un costrutto while: ogni letterale presente nella coda LIFO viene confrontato con la query, se i due corrispondono viene restituito 'True', altrimenti si prosegue. L'inferenza consiste nel verificare se il letterale in esame è presente nelle permesse delle regole di derivazione della base di conoscenza, in tal caso il numero di letterali coinvolti nella premessa viene ridotto di uno. Quando viene raggiunto lo zero, la conseguenza è inferita, ovvero assume valore di verità 'True'. Il processo è iterato finchè non viene inferita la query, altrimenti viene restituito 'False'.

Tra i parametri in ingresso viene inoltre passata una lista nella quale verranno memorizzati i letterali analizzati in ordine temporale. Sarà utile per il confronto tra gli algoritmi.

3.3.2 Backward Chaining Entailment

L'algoritmo di backward chaining viene qui implementato in forma ricorsiva. Una funzione **BCEntailment** prende come parametri la base di conoscenza e la query in esame. Un dizionario denominato 'inferred', tiene traccia dei letterali inferiti. Tale dizionario, la base di conoscenza e la query vengono passati come parametri d'ingresso alla funzione **BCEntails**. É in quest'ultima funzione che viene applicato il processo di inferenza. Viene subito valutato se la query è stata inferita e se ha valore di verità corrispondente a 'True', in tal caso viene ritornato il valore 'True' che significa che l'inferenza è riuscita. Altrimenti, la query viene confrontata con le conclusioni delle regole in esame, se le premesse sono vere, allora avviene l'inferenza della query, altrimenti viene ricorsivamente applicato l'algoritmo anche ai letterali che costituisono le premesse.

Inoltre, tra i parametri in ingresso, viene passata una lista nella quale verranno memorizzati i letterali analizzati in ordine temporale. Sarà utile per il confronto tra gli algoritmi.

3.4 Dati e risultati

Nel programma sono stati fatti quattro test per verificare la correttezza degli algoritmi e poterli mettere a confronto. Ogni test prevede la definizione dei letterali coinvolti e delle regole di derivazione. Tra i letterali sono presenti sia i simboli da inferire sia i fatti conosciuti, cui viene assegnato fin da principio il loro valore di verità. I simboli e le regole vengono raccolti in due liste, le quali costituiranno la base di conoscenza. Le basi di conoscenza e le queries prese in esame sono le seguenti:

	Regole		Fatti	Query
	Premesse	Conclusione		
TEST 1	nB11	nP12	nB11 = True	nP12
	nB11	nP12		
TEST 2	nB11	nP12	nB11 = True	P12
	nB11	nP12		
TEST 3	nS12, S21	nW22	nS12 = True	W31
	S21, nW22	W31	S21 = True	
TEST 4	nS12, S21	nW22	nS12 = True	nW31
	S21, nW22	W31	S21 = True	

Ogni query viene sottoposta al procedimento di inferenza con entrambi gli algoritmi. Di seguito i risultati:

	Risultato FC	Ordine d'esplorazione FC	Risultato BC	Ordine d'esplorazione BC
TEST 1	True	[nB11, nP21, nP12]	True	[nB11]
TEST 2	False	[nB11, nP21, nP12]	False	[]
TEST 3	True	[S21, nS12, nW22, nW31]	True	[S21, nW22]
TEST 4	False	[S21, nS12, nW22, nW31]	False	[]

Grazie alla lista in cui vengono memorizzati i letterali analizzati durante l'esecuzione dell'algoritmo, è possibile osservare i due diversi metodi di inferenza. I risultati ottenuti corrispondono a quelli desiderati.

É facile osservare che l'algoritmo di backward chaining esplora solo i rami necessari ad ottenere inferenza. Infatti, prendendo ad esempio i test in cui si sottopone a inferenza una query falsa, l'algoritmo verifica subito che questa non rientra tra le conclusioni di alcuna regola, restituendo così il valore 'Falso' senza eseguire alcun processo di inferenza.

3.5 Codice e fonti

Per poter osservare i risultati del programma è sufficiente eseguire il file main.py contenuto nel progetto AI-project.

La fonte utilizzata per attingere lo pseudo-codice di uno degli algoritmi e la descrizione del wumpus world è il libro "Artificial Intelligence: A Modern Approach".