Report sull'Algoritmo di Variable Elimination per Reti Bayesiane

Simone Riccio

28 giugno 2025

Sommario

Questo documento descrive l'implementazione dell'algoritmo di Variable Elimination per il calcolo delle probabilità marginali in una rete bayesiana. L'obiettivo è fornire una panoramica dell'algoritmo, delle scelte implementative in C++ e di un esempio pratico di utilizzo.

Indice

1 Introduzione

Le reti bayesiane sono modelli grafici probabilistici che rappresentano un insieme di variabili aleatorie e le loro dipendenze condizionali tramite un grafo aciclico diretto (DAG). Il calcolo di inferenza, come la determinazione della probabilità marginale di una variabile data un'evidenza, è un compito fondamentale ma computazionalmente complesso.

2 L'Algoritmo di Variable Elimination

L'algoritmo di Variable Elimination (VE) è una tecnica per calcolare esattamente le probabilità marginali in una rete bayesiana. L'idea centrale è di eliminare le variabili non di interesse una alla volta, sommando (o marginalizzando) su di esse.

2.1 Fattori

L'algoritmo opera su una struttura dati chiamata "fattore". Un fattore $\phi(X_1, \ldots, X_k)$ è una funzione che mappa ogni possibile assegnazione di valori alle variabili $X_1, \ldots, X_k$ a un numero reale non negativo. Le tabelle di probabilità condizionale (CPT) della rete sono i fattori iniziali.

2.2 Passi dell'Algoritmo

L'algoritmo si articola nei seguenti passi:

1. **Inizializzazione:** Per ogni variabile, si crea un fattore corrispondente alla sua CPT.
2. **Riduzione dell'evidenza:** Se sono presenti variabili di evidenza, si riducono i fattori per riflettere i valori osservati.
3. **Eliminazione delle variabili:** Per ogni variabile da eliminare (che non è né di query né di evidenza):
   (a) Si raccolgono tutti i fattori che includono la variabile.
   (b) Si calcola il prodotto di questi fattori.
   (c) Si marginalizza la variabile dal fattore prodotto, creando un nuovo fattore.
4. **Risultato finale:** Dopo aver eliminato tutte le variabili necessarie, si calcola il prodotto dei fattori rimanenti.
5. **Normalizzazione:** Il risultato viene normalizzato per ottenere una distribuzione di probabilità.

3 Implementazione in C++

La nostra implementazione in C++ si concentra sulla creazione di una classe `Factor` flessibile e sull'orchestrazione dei passi dell'algoritmo VE.

3.1 Struttura Dati per i Fattori

Un fattore è stato implementato come una classe che contiene:

- Un elenco delle variabili coinvolte.

- Una mappa o un vettore per memorizzare i valori del fattore per ogni assegnazione.

```
class Factor {
public:
    // Costruttori, metodi per il prodotto, la marginalizzazione, etc.

private:
    std::vector<std::string> variables;
    std::map<std::vector<int>, double> values;
};
```

Listing 1: Esempio di scheletro della classe Factor in C++.

4 Esempio di Esecuzione

Consideriamo una semplice rete bayesiana (es. la rete "Sprinkler"). Qui si può inserire un'immagine della rete.

Si mostra l'output del programma per una query specifica, ad esempio $P(\text{WetGrass}|\text{Sprinkler} = \text{true})$.

5 Conclusioni

L'algoritmo di Variable Elimination si è dimostrato efficace per il calcolo di inferenza esatta. L'implementazione in C++ ha richiesto un'attenta gestione delle strutture dati, in particolare per la rappresentazione e la manipolazione dei fattori.