UNIVERSITÀ
DEGLI STUDI
FIRENZE

Apprendimento dei Parametri in Reti Bayesiane

Analisi della Convergenza tramite Divergenza di Jensen-Shannon

Simone Suma

Anno Accademico 2024/2025

1. Introduzione e Scopo del Progetto

Le Reti Bayesiane rappresentano uno strumento fondamentale nell'ambito dei modelli grafici probabilistici, consentendo di modellare relazioni complesse tra variabili aleatorie e di ragionare in condizioni di incertezza. Tali reti si basano su una struttura a grafo aciclico diretto (DAG), dove ciascun nodo rappresenta una variabile, e gli archi codificano relazioni di dipendenza causale o statistica.

Nel presente lavoro si assume che la struttura della rete (DAG) sia nota a priori: l'obiettivo è quindi limitato all'apprendimento dei parametri, ovvero delle distribuzioni di probabilità condizionata (CPT) per ciascun nodo. In altre parole, si vuole stimare il comportamento probabilistico della rete a partire da dati osservati, mantenendo fissa la topologia del grafo.

L'obiettivo di questo progetto è duplice. In primo luogo, implementare un algoritmo robusto per l'apprendimento dei parametri di una rete Bayesiana, basato su un approccio Bayesiano con priori non informative, utilizzando lo smoothing di Laplace per gestire i casi di bassa frequenza. In secondo luogo, verificare sperimentalmente una delle proprietà teoriche fondamentali dell'apprendimento statistico: la convergenza della stima al modello reale all'aumentare dell'evidenza empirica.

2. Metodologia

L'esperimento è stato condotto seguendo una pipeline ben definita per garantire la riproducibilità dei risultati.

2.1 Reti di Riferimento

L'esperimento è stato condotto utilizzando due diverse reti Bayesiane come ground truth per valutare la robustezza e la scalabilità dell'algoritmo di apprendimento.

La prima rete utilizzata è la **Asia**, un modello piccolo e semplice, caricata dal file `asia.bif`. Questa rete ha una struttura ridotta ed è spesso impiegata per test preliminari o per illustrare i concetti fondamentali delle Reti Bayesiane.

La seconda rete utilizzata è la **Andes**, un modello complesso, ampiamente usato per la valutazione delle conoscenze degli studenti in ambito fisico. Questa rete, caricata dal file `andes.bif`, presenta una struttura ampia e articolata, rappresentando un benchmark impegnativo.

L'utilizzo di entrambe le reti ha permesso di confrontare le prestazioni dell'algoritmo su modelli di complessità diversa e di validarne l'efficacia sia in scenari semplici che complessi.

2.2 Generazione dei Dati

Per ogni dimensione del campione n testata (da 50 a 10.000), è stato generato un dataset di training tramite **campionamento ancestrale** a partire dalla rete di riferimento p. Questo processo assicura che i dati riflettano la distribuzione di probabilità vera.

2.3 Apprendimento dei Parametri

Per ogni dataset generato, i parametri della rete (le CPT) sono stati stimati da zero utilizzando un approccio Bayesiano. In particolare, si è applicato lo smoothing di Laplace (o "add-one

smoothing"), che consiste nell'aggiungere un piccolo conteggio a ogni possibile evento, anche a quelli non osservati nel campione. Questo evita che venga assegnata una probabilità nulla a eventi mai verificatisi nel dataset, rendendo le stime più robuste e realistiche.

2.4 Misura dell'Errore

La "distanza" tra la distribuzione vera p e quella appresa q_n è stata quantificata calcolando la media della **Divergenza di Jensen-Shannon (JS)** su tutte le distribuzioni condizionali (CPT) della rete. In dettaglio, per ogni nodo si calcola la divergenza JS tra la distribuzione condizionale vera e quella stimata, e si prende la media di questi valori.

La formula della divergenza JS tra due distribuzioni discrete p e q_n su uno spazio di configurazioni U è:

$$JS(p, q_n) = \frac{1}{2} \sum_{u \in U} p(u) \log \frac{p(u)}{M(u)} + \frac{1}{2} \sum_{u \in U} q_n(u) \log \frac{q_n(u)}{M(u)}$$

dove $M(u) = \frac{p(u)+q_n(u)}{2}$. Un valore di JS pari a 0 indica una corrispondenza perfetta tra le distribuzioni.

3. Risultati Sperimentali

L'esperimento è stato eseguito per diverse dimensioni del campione n. Per ciascuna dimensione, è stato calcolato l'errore medio tra la rete vera e quella appresa tramite la Divergenza JS. I risultati numerici sono riportati nella Tabella 1.

Numero di Campioni (n)	**Divergenza JS Media**
50	0.0454
100	0.0354
250	0.0206
500	0.0135
1000	0.0081
2500	0.0047
5000	0.0027
10000	0.0016

Tabella 1: Errore di stima (Divergenza JS) in funzione della dimensione del campione.

Questi dati sono visualizzati nella curva di apprendimento in Figura 1. Il grafico utilizza una scala logaritmica per l'asse X per rappresentare in modo efficace l'ampio intervallo di dimensioni del campione testate.

Dal grafico si evince una relazione inversa, fluida e monotona tra il numero di campioni e l'errore di stima. Il calo dell'errore è particolarmente marcato nella fase iniziale (da 50 a 1000 campioni), indicando che i maggiori guadagni in termini di accuratezza si ottengono con i primi dati a disposizione. Superata tale soglia, ulteriori aumenti nella dimensione del campione portano a miglioramenti più marginali, con la stima che tende asintoticamente al modello vero (errore zero), mostrando una convergenza robusta e consistente.

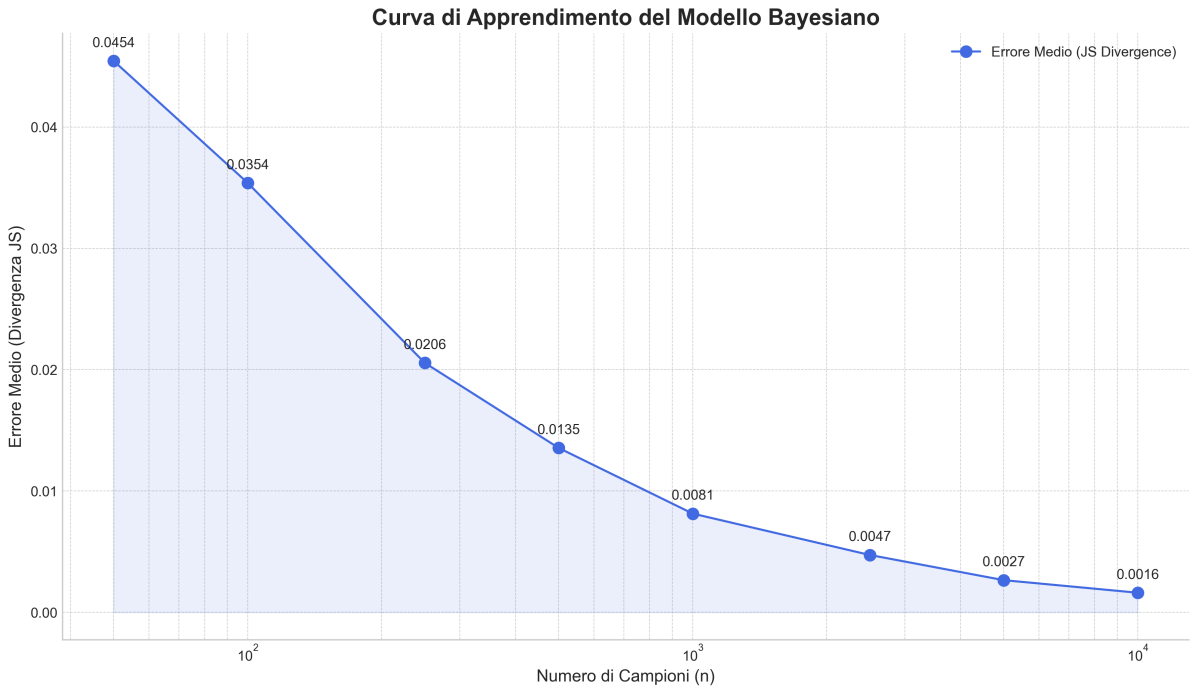

Figura 1: Curva di apprendimento che mostra la Divergenza JS media in funzione del numero di campioni.

4. Dettagli Implementativi

Il progetto è stato sviluppato in Python 3.10.5, con un'attenzione particolare alla modularità del codice, alla scelta di strutture dati efficienti e alla robustezza degli algoritmi.

4.1 Struttura del Codice

Il codice è suddiviso in tre file principali per separare le responsabilità:

- `rete_bayesiana.py`: Definisce le classi `Nodo` e `ReteBayesiana`. Questa astrazione permette di rappresentare la rete in modo orientato agli oggetti, incapsulando la logica di parsing e di generazione dei dati.
- `algoritmi.py`: Contiene le funzioni "pure" che implementano la logica matematica, come l'apprendimento dei parametri e il calcolo della divergenza. Questa separazione rende gli algoritmi indipendenti dalla specifica rappresentazione dei dati.
- `main.py`: Funge da orchestratore, gestendo il flusso dell'esperimento e la visualizzazione dei risultati.

4.2 Scelta delle Strutture Dati

La performance e la chiarezza del codice dipendono in modo critico dalle strutture dati adottate:

- **Rappresentazione della Rete:** La rete è implementata come un dizionario che associa i nomi dei nodi (stringhe) ai rispettivi oggetti `Nodo`. Questa scelta garantisce un accesso ai nodi in tempo costante O(1), fondamentale per un'efficiente navigazione del grafo.

- **Tabelle di Probabilità Condizionata (CPT):** Le CPT sono implementate tramite dizionari nidificati. Per i nodi con genitori, le chiavi del dizionario principale sono delle **tuple** che rappresentano tutte le combinazioni possibili degli stati dei genitori. Le tuple sono preferite rispetto alle liste perché immutabili e quindi utilizzabili come chiavi di dizionario, consentendo un accesso diretto e performante alla distribuzione di probabilità corretta per ogni configurazione di evidenza.

4.3 Ordinamento Topologico dei Nodi

Per garantire il corretto funzionamento degli algoritmi di campionamento, è essenziale processare i nodi in un ordine in cui i genitori precedano sempre i figli. Questo perché, per calcolare la distribuzione di probabilità condizionata di un nodo figlio, è necessario conoscere i valori assegnati ai suoi genitori, che influenzano direttamente la sua distribuzione. Processare prima i genitori assicura che, al momento della valutazione del nodo figlio, tutte le informazioni rilevanti siano già disponibili, mantenendo così la coerenza del modello probabilistico.

4.4 Campionamento Ancestrale

La generazione dei dati di training è stata realizzata tramite il metodo del **campionamento ancestrale**. L'algoritmo, implementato in `genera_campione`, opera come segue:

1. I nodi della rete vengono processati secondo l'ordine topologico.
2. Per ogni nodo, si seleziona la distribuzione di probabilità appropriata dalla sua CPT, accedendovi tramite i valori dei suoi nodi genitori, già campionati nei passi precedenti.
3. Viene estratto un valore casuale per il nodo corrente utilizzando la funzione `random.choices`, che permette di effettuare una scelta pesata basata sulla distribuzione di probabilità selezionata.

Questo processo assicura che ogni campione generato sia una realizzazione coerente e statisticamente valida della distribuzione di probabilità congiunta definita dalla rete.

4.5 Apprendimento Bayesiano

La funzione `impara_parametri` implementa lo smoothing di Laplace con una strategia a due fasi per garantire la robustezza nell'apprendimento dei parametri.

- **Fase 1: Inizializzazione dei Conteggi.** Prima di analizzare i dati, la funzione precalcola tutte le possibili combinazioni degli stati dei nodi genitori utilizzando `itertools.product`. Per ciascuna combinazione, viene creata una struttura di conteggio in cui ogni possibile stato del nodo figlio è inizializzato a 1, applicando così una correzione che evita che probabilità possano risultare zero per eventi mai osservati.
- **Fase 2: Aggiornamento e Normalizzazione.** Successivamente, il dataset viene iterato una sola volta. Per ogni campione, viene identificata la condizione dei genitori e il risultato del nodo figlio, e il contatore corrispondente viene incrementato. Infine, i conteggi vengono normalizzati per ottenere le probabilità stimate.

Questo approccio previene errori di tipo `KeyError` dovuti a combinazioni di evidenza non presenti nel dataset di training e garantisce che la CPT appresa abbia sempre la stessa struttura di quella originale.

5. Analisi e Conclusioni

L'esperimento condotto ha confermato con successo l'ipotesi centrale del progetto: la precisione di un modello Bayesiano appreso dai dati converge verso quella del modello reale all'aumentare della quantità di evidenza disponibile. La curva di apprendimento, ottenuta sulla complessa rete *Andes*, mostra chiaramente una relazione inversa e monotona tra la dimensione del campione e l'errore di stima, misurato tramite la Divergenza di Jensen-Shannon. Questo risultato non è una semplice constatazione numerica, bensì una dimostrazione pratica di un principio fondamentale dell'intelligenza artificiale moderna: la capacità di generalizzare e costruire modelli accurati del mondo dipende intrinsecamente dalla quantità e qualità dei dati.

Le scelte implementative si sono rivelate adeguate alla complessità del problema. L'adozione di un algoritmo di ordinamento topologico generico ha garantito la flessibilità del software, mentre la strategia di pre-inizializzazione dei conteggi nella fase di apprendimento ha assicurato la robustezza del modello anche in presenza di combinazioni di eventi non osservate nei dataset più piccoli. In particolare, lo smoothing di Laplace si è dimostrato essenziale per evitare il problema delle probabilità nulle, un fenomeno che potrebbe rendere un modello probabilistico inutilizzabile.

È tuttavia importante riconoscere i limiti di questo lavoro, che rappresentano spunti per futuri sviluppi.L'assunzione di una struttura di rete nota a priori rappresenta una semplificazione significativa: in scenari reali, spesso la struttura è sconosciuta e deve essere scoperta dai dati attraverso metodi appositi. Inoltre, l'assunzione di dati completi è un altro limite rilevante, poiché nella pratica i dataset contengono frequentemente valori mancanti. Per affrontare queste problematiche, sarebbe opportuno sviluppare metodi in grado di identificare automaticamente le relazioni tra variabili e di gestire l'incertezza dovuta a dati parziali o incompleti, rendendo così il modello più robusto e applicabile a contesti reali.

In conclusione, il progetto ha pienamente raggiunto i suoi obiettivi, offrendo non solo un'implementazione funzionante e robusta dell'apprendimento bayesiano, ma anche una comprensione pratica approfondita dei meccanismi di convergenza che regolano i modelli statistici.

6. Riferimenti Bibliografici e Fonti

Riferimenti bibliografici

[1] D. Heckerman, *A Tutorial on Learning with Bayesian Networks*, Microsoft Research, Technical Report MSR-TR-95-06, 1995 (revised 1997).

[2] S. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed., Pearson

[3] N. Cullen, *pyBN GitHub Repository*: `https://github.com/ncullen93/pyBN`. Le reti `asia.bif` e `andes.bif` sono state prelevate dalla directory `/data` di questo repository.