Apprendimento di alberi di decisione

Simone Pezzulla

Settembre 2020

1 Introduzione

Nell'elaborato è stato sviluppato il codice in Python per l'apprendimento degli alberi di decisione discusso in classe e riferito in RN 2009 §18.3, utilizzando l'entropia come misura di impurità. Successivamente è stato implementato un metodo di pruning sulle regole corrispondenti ai vari cammini dell'albero e sono stati confrontati i risultati sul dataset Adult.

2 Dettagli tecnici

Il progetto è stato svolto su *JupyterNotebook* con l'ausilio delle seguenti librerie esterne:

- **pandas** per la lettura e gestione del dataset
- **sklearn** per la divisione casuale del dataset in tre parti: training-set, validation-set e test-set, dove l'ultimo ha dimensione del 20% e il restante è spartito rispettivamente con percentuali del 67% e 33%

Inoltre sono state eseguite alcune semplificazioni al dataset come la discretizzazione di valori continui e l'eliminazione di alcuni dati contenenti valori non utilizzabili. Tali operazioni sono visibili in dettaglio all'interno del codice. Per quanto riguarda le terminologia, si interà per *rule* (o *regola*) un'accoppiata attributo-valore e per *rules* un insieme di accoppiate che rappresenta un cammino dell'albero (tuttavia le notazioni nel codice in python differiranno).

3 Implementazione

Ogni cammino dalla radice alle foglie viene salvata su una lista, sulla quale verranno effettuati in base alle esigenze dei test prima, durante e dopo il pruning. Le funzioni di testing sono le seguenti:

- $test\_dtree(dtree)$ Restituisce un punteggio tra 0 e 1 dato da

$$score = \frac{p}{t}$$

dove p è il numero di valori indovinati dell'albero sul test-set e t è il numero totale di dati del test-set

- $test\_rules(dataset, rules)$ Effettua su tutta la lista di $rules$ lo stesso procedimento della funzione recedente
- $errors(rule)$ e $score(rule)$ Eseguono il test su un insieme di $rules$ e forniscono rispettivamente il numero di errori prodotta da essa e un punteggio

3.1 Algoritmo di pruning

Il pruning si basa sull'errore del validation set. Si prendono in considerazione le $rules$ dove vi è maggiore quantità di regole poiché c'è maggiore probabilità di $overfitting$. Viene quindi tolta la $rule$ che permette di ottenere alle rimanenti $rule$ il punteggio migliore senza di essa. Tale punteggio verrà verificato dalla funzione $score(rule)$. Successivamente si confronta il numero n di errori della nuova $rules$ e s, dato da

$$s = \sum_{srules} error(srules)$$

dove $srules$ sono l'insieme di $rules$ che interessano gli stessi cammini della nuova $rules$. Se $n \geq s$ e pertanto il numero di errori è aumentato o rimasto invariato dopo il pruning, si torna allo stato iniziale, dopodiché si itera il procedimento su un'altra rule. L'algoritmo si arresta se non ci sono più $rule$ disponibili per il pruning o se viene raggiunto un limite di lunghezza minimo della $rule$ oltre al quale non è possibile effettuare il pruning.

4 Risultati

Su diverse iterazioni il punteggio dell'albero di decisione si attesta intorno a 0.84-0.87. Il punteggio del post pruning invece varia molto a seconda del limite imposto. In generale si nota che più basso è il limite e più si abbassa la precisione delle nuove regole.

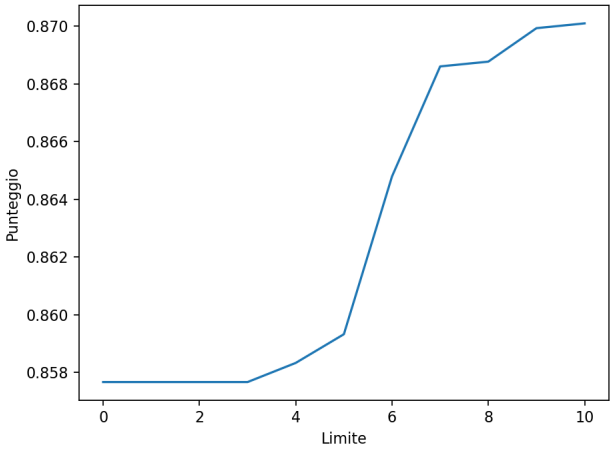

Figure 1: Iterazioni di pruning sulla stessa lista di *rules* ma variando il limite. Il punteggio iniziale senza pruning è di 0.8592