Alberi di Decisione

Tommaso Bucaioni

1 Introduzione

Il progetto prevede lo sviluppo di codice per l'apprendimento di alberi di decisione e l'implementazione di una tecnica di pruning. Gli alberi di decisione vengono usati nel campo dell'intelligenza artificiale per classificare una serie di esempi sulla base di osservazioni effettuate da parte di un agente. Un albero di decisione prende in ingresso delle coppie esempi-target e ritorna un classificatore che cercherà di predire il target dei prossimi input. In un albero di decisione i nodi rappresentano i test che vengono effettuati su un determinato attributo e ogni ramo rappresenta un valore di quell'attributo. La foglia invece rappresenta il valore predetto dall'albero.

2 Algoritmo

2.1 Implementazione

L'algoritmo per l'apprendimento utilizzato è quello descritto in R&N 2009 §18.3. L'organizzazione delle classi e alcune delle funzioni presenti sono state riprese da aima-python repository. Sono state create 3 classi:

- DataSet.py: definisce il dataset con i suoi esempi, attributi, valori e il target
- DecisionTree.py: rappresenta la struttura dell'albero con i (nodi) attributi e i valori (rami)
- DecisionLeaf.py: rappresenta la foglia contenente il target

2.2 Funzionamento

L'algoritmo è di tipo ricorsivo e prende in ingresso la lista di esempi, gli attributi e la lista di esempi derivanti dal test precedente (inizialmente vuota). Inizialmente testa le seguenti condizioni:

- se all'i-esima ricorsione gli esempi sono terminati la funzione *plurality_value* ritorna la classe più comune rispetto all'insieme *parent_examples*

- se all'i-esima ricorsione gli esempi contengono tutti la stessa classificazione l'algoritmo ritorna una foglia contenente quella classe come target.
- se all'i-esima ricorsione sono finiti gli attributi da testare l'algoritmo ritorna *plurality_value* rispetto agli esempi rimasti
- se all'i-esima ricorsione si verifica la condizione *percent_error(examples)* < *error_threshold* viene attuata la strategia di pruning che viene spiegata al paragrafo §2.4

Se non viene verificata una delle precedenti condizioni l'algoritmo seleziona l'attributo migliore tra quelli disponibili e costruisce un albero avente come radice l'attributo scelto. Successivamente viene chiamata ricorsivamente la funzione *decison_tree_learning* che crea un sottoalbero per ogni valore che quell'attributo può prendere. Infine aggancia i sottoalberi creati al nodo radice.

2.3 Scelta dell'attributo

Abbiamo detto precedentemente che viene scelto l'attributo migliore tra quelli disponibili. L'importanza di un attributo si basa sul concetto di information gain che a sua volta si basa sull'entropia. L'entropia è la misura di incertezza di una variabile aleatoria cioè per una variabile V con valori v_k , ognuno con probabilità $P(v_k)$ l'entropia è definita:

$$H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)}$$

Sulla base dell'entropia, l'information gain invece è la riduzione di entropia che ci si aspetta quando si sceglie un attributo su cui fare il test. Verrà scelto l'attributo con l'information gain più alto, cioè quello che ci fornisce maggiore informazione sul target. Formalmente sia T un insieme di esempi del tipo $(x, y) = (x_1, x_2, ..., x_k, y)$ dove x_i con $x_a \in vals(i)$ è il valore dell'i-esimo attributo e y è il target da predire, l'information gain ha la seguente forma:

$$Gain(T, a) = H(T) - \sum_{v \in vals(a)} \frac{|\{x \in T | x_a = v\}|}{|T|} \cdot H(\{x \in T | x_a = v\})$$

2.4 Tecnica di pruning

I 3 datasets analizzati vengono divisi in training set, validation set e test set secondo le proporzioni definite dai parametri inseriti nella funzione *train_val_test*. Il training set viene usato per costruire l'albero di decisione, il validation set per testare la qualità dell'albero di decisione creato. Successivamente effettueremo i test sul test set. Per evitare problemi di **overfitting** viene adottata una semplice strategia di pre-pruning basata sull'errore sul validation set. La funzione *percent_error(examples)* calcola il numero di esempi che hanno la classe di maggioranza e ritorna l'errore percentuale che noi commetteremmo se generassimo una foglia con quella classe di maggioranza. Questa percentuale viene confrontata con il parametro *error_threshold* e se è minore genera una foglia contenente la classe di maggioranza rispetto agli esempi testati.

3 Test

3.1 Dataset

I dataset usati sono ripresi da UCI Machine Learning Repository. In particolare il dataset *car-evaluation* contiene 1728 esempi, *balance-scale.txt* contiene 625 esempi e *tic-tac-toe.txt* contiene 958 esempi.

3.2 modalità di test

I test vengono effettuati tramite le funzioni presenti in *test.py*. In esso sono presenti le funzioni per creare il dataset, suddividerlo in training set, validation set e test set (60%, 20%, 20%) e calcolare l'accuratezza. Nel file *main.py*, dopo aver creato e diviso il dataset, è presente un ciclo for che ad ogni iterazione:

- crea un albero di decisione con valore di *error_threshold* crescente (da 0 a 100 con passo 1).
- effettua i test su training, validation e test e li inserisce rispettivamente in *train_results, val_results* e *test_results* (serviranno per creare il grafico dei risultati)

Tramite l'uso delle funzioni della libreria *matplotlib.pyplot* viene creato un grafico contenente l'accuratezza di training set, validation set e test set al variare di *error_threshold*

4 Presentazione risultati

4.1 Car evaluation

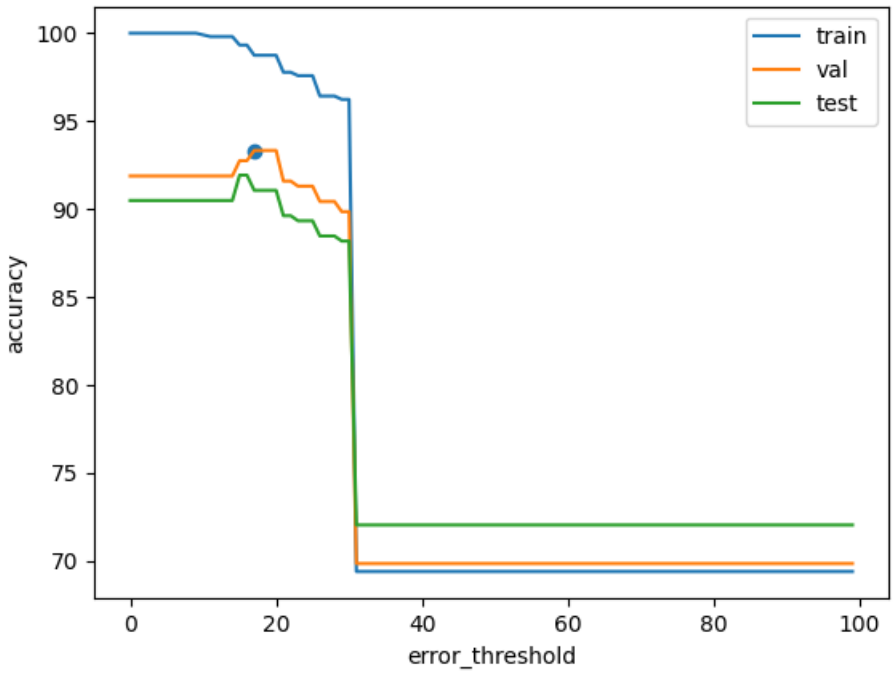
Il grafico sottostante riporta i risultati effettuati sul dataset *car-evalutaion.txt*

Dall'immagine possiamo osservare come una diminuzione dell'accuratezza sul training set vada a migliorare l'accuratezza sul validation set. Di conseguenza i test effettuati sul test set migliorano rispetto a quelli effettuati sullo stesso set senza fare pruning. Più precisamente, i dati ricavati dall'analisi effettuata su questo test sono i seguenti:

	Senza pruning	con pruning*
accuratezza sul training set	100%	98.3%
accuratezza sul validation set	91.8%	93.3%
accuratezza sul test set	90.4%	91%

*i valori rappresentano le percentuali di accuratezza calcolate quando l'accuratezza sul validation set è massima

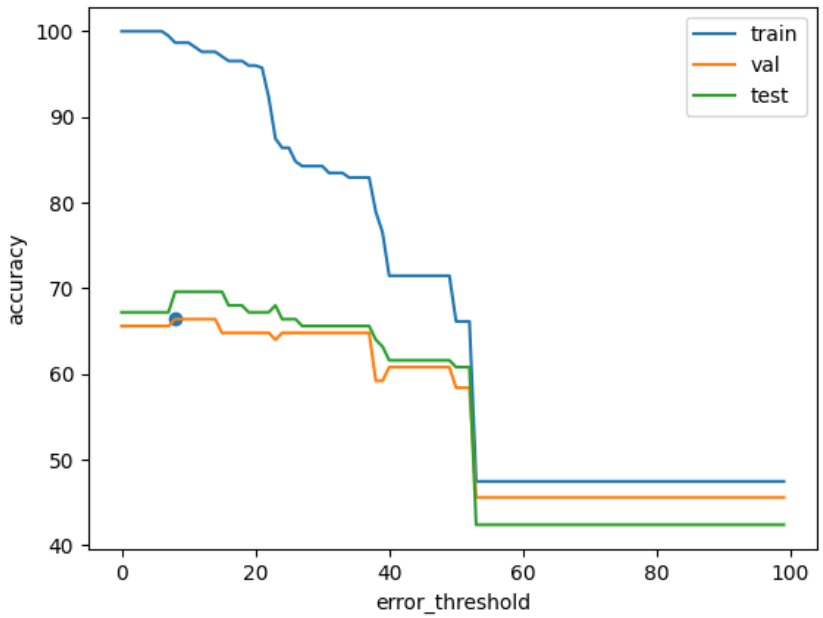
In particolare la tecnica di pruning ci consente di migliorare l'accuratezza sul test set di 0.6 punti percentuali.



La figura mostra l'accuratezza su training set, validation set e test set in funzione di entropy_threshold. In rosso viene evidenziato il punto in cui l'accuratezza sul validation set è massima

4.2 Balance scale

Il grafico sottostante riporta i risultati effettuati sul dataset *balance-scale.txt*.



La figura mostra l'accuratezza su training set, validation set e test set in funzione di entropy_threshold. In rosso viene evidenziato il punto in cui l'accuratezza sul validation set è massima

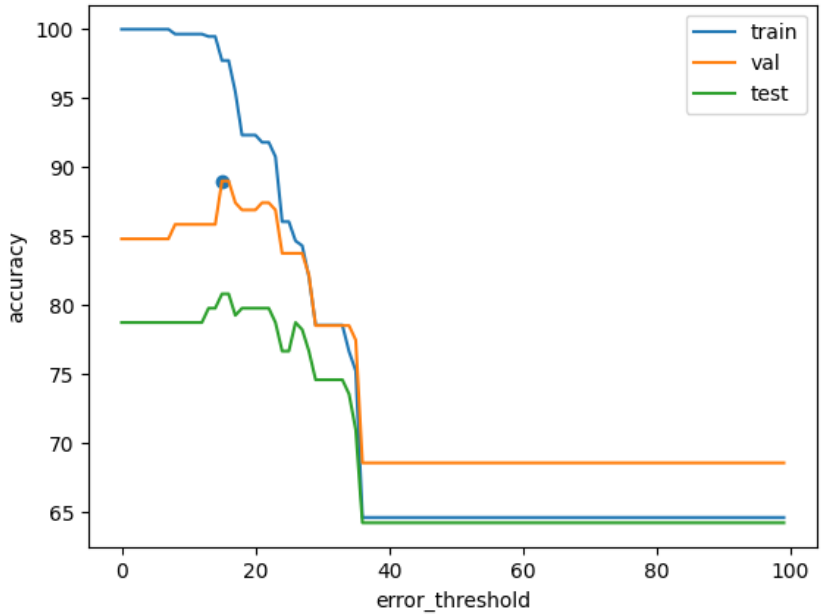
	Senza pruning	con pruning*
accuratezza sul training set	100%	98.6%
accuratezza sul validation set	65.6%	66.4%
accuratezza sul test set	67.2%	69.6%

*i valori rappresentano le percentuali di accuratezza calcolate quando l'accuratezza sul validation set è massima

In particolare la tecnica di pruning ci consente di migliorare l'accuratezza sul test set di 2.4 punti percentuali.

4.3 Tic tac toe

Il grafico sottostante riporta i risultati effettuati sul dataset *tic-tac-toe.txt*.



La figura mostra l'accuratezza su training set, validation set e test set in funzione di entropy_threshold. In rosso viene evidenziato il punto in cui l'accuratezza sul validation set è massima

	Senza pruning	con pruning*
accuratezza sul training set	100%	97.7%
accuratezza sul validation set	84.8%	89%
accuratezza sul test set	78.8%	80.8%

*i valori rappresentano le percentuali di accuratezza calcolate quando l'accuratezza sul validation set è massima

In particolare la tecnica di pruning ci consente di migliorare l'accuratezza sul test set di 2.1 punti percentuali.

5 Conclusioni

In tutti i 3 dataset scelti la tecnica di pruning ci consente di avere un miglioramento sulla predizione del target nel test set. In particolare nel dataset *car_evaluation.txt* la percentuale è molto vicina al 100%: questo è dovuto al numero significativo di istanze (1728) che ci consentono di costruire un albero che riesce a predire bene il target. Negli altri due casi, soprattutto il secondo, la mancanza di un numero sufficientemente grande di esempi non ci permette di avere un'accuratezza tanto alta.