Decision Tree Learning

Assioma Andrea Aligi

5 febbraio 2024

1. Introduzione

Il seguente elaborato mostra la realizzazione di un albero decisione partendo da quello descritto nel capitolo 19.3 del libro Aritficial Intelligence: A Modern Approach (di Russell e Norvig (edizione 2020)), al quale viene modificato il passo base dello pseudocodice in modo che la ricorsione viene interrotta se la profondità è maggiore di un intero P assegnato oppure se il numero di esempi in examples è minore di un intero M assegnato.

Lo pseudocodice preso in considerazione sfrutta i concetti di entropia e di information gain come metodi di inserimento dei nodi all'interno dell'albero, entrambi della teoria dell'informazione.

1.1. Hardware e sistema operativo utilizzati

Il codice è stato eseguito su un pc fisso con processore i5-10600k e 32gb di ram. Il sistema operativo usato è Arch Linux.

2. Creazione dell'albero di decisione

Per realizzare l'albero di decisione viene estratto il dataset dalla libreria python `ucimlrepo` tramite id (per esempio l'iris ha come id il 53). Successivamente, viene eseguito uno shuffle per randomizzare gli elementi all'interno: essendo gli esempi ordinati per target, ciò permette di dividere training set e test set in modo da evitare che quest'ultimo abbia istanze di un solo target.
Una volta ottenuti i due dataset, viene chiamata la funzione che realizza l'albero: ricorsivamente, vengono analizzati tutti gli esempi in modo tale da determinare quale attributo inserire prima come nodo dell'albero, oppure viene aggiornata la frontiera aggiungendo foglie all'albero.
Ottenuto l'albero di decisione, viene disegnato graficamente tramite l'utilizzo della libreria `networkx` e ne viene calcolata la precisione testando sia il training dataset che il test dataset.

I dataset utilizzati sono stati reperiti da UCI Machine Learning Repository:

- Iris: classifica le tipologie di iris in base a una serie di attributi. Le variabili sono `petal width`, `sepal width`, `petal length`, `sepal length` e `class` (setosa, versicolor, virginica).

- Heart failure: `age`, `anemia`, `creatinine_phosphofinase` (indica il livello dell'enzima CPK nel sangue), `diabetes`, `ejection_fraction` (percentuale di sangue che esce dal cuore a ogni contrazione), `high_blood_pressure`, `platelets`, `serum_creatinine` (livello di siero di creatinine nel sangue), `serum_sodium` (livello di siero di sodio nel sangue), `sex`, `smoking`, `time` (periodo di follow-up), `death_event` (se il paziente muore durante il periodo follow-up (target)).
- Wine: `Alcohol`, `Maliacid`, `Ash`, `Alcalinity_of_ash`, `Magnesium`, `Total_phenols`, `Flavanoids`, `Nonflavanoid_phenols`, `Proathocyanins`, `Color_intensity`, `Hue`, `OD280_OD315_of_diluted_wines`, `Proline`, `class` (tipo di vino).

Per la realizzazione dell'albero, sono stati realizzati 5 files differenti:

- `main.py`: chiede in input l'id del dataset, lo split train/test, la massima profondità dell'albero (`P`) e il numero minimo di examples (`M`). Successivamente, chiama la funzione che crea l'albero, lo disegna e ne stampa la precisione (`create_tree`).
- `tree_creation.py`: definisce la funzione `create_tree` in cui vengono inizializzati i dataset (training e test) e creato l'albero di decisione tramite l'istanziazione di una variabile `DecisionTree`. Vengono inoltre calcolate e stampate le precisioni tramite la chiamata alla funzione `test_tree`, e viene disegnato e rappresentato graficamente (grazie alla libreria `matplotlib`) l'albero tramite la chiamata alla funzione `plot_tree`.
- `decision_tree.py`: definisce la classe `DecisionTree`.
- `tree_elements.py`: definisce le classi `Branch` e `Node` necessarie per la realizzazione dell'albero di decisione.
- `plot_tree.py`: definisce le funzioni che disegnano l'albero di decisione.
- `tree_testing.py`: definisce le funzioni che permetto il calcolo della precisione dell'albero (`test_tree` e `tree_precision`).

3. Analisi dei risultati

Nota: per scegliere `M` è stata analizzata la cardinalità degli insiemi di esempi dei sottoalberi del decision tree

3.1. Dataset Iris

Nel caso del dataset Iris si hanno i seguenti risultati:

Split	P	M	training precision	test precision
30/150	3	0	0.8	0.7
30/150	2	0	1.0	0.8667
30/150	1	0	0.433333	0.3333
30/150	3	50	0.43333	0.3333
30/150	3	3	0.66667	0.5
30/150	3	2	0.86667	0.6667
30/150	2	2	0.9333	0.66667

Dalla seguente tabella si possono notare i seguenti comportamenti:

- nei casi P=1 M=0 e P=3 M=50 accade la stessa cosa in quanto vengono eliminati gli stessi sotto-esempi, le cui dimensioni variano tra [1,11]. Perciò si forma sempre un albero di profondità 1.
- ridurre di 1 la profondità dell'albero ne aumenta la precisione sia nel caso del training che nel caso del test; aumentare il limite minimo dell'insieme di esempi diminuisce la precisione.
- M influenza soprattutto la precisione del training (casi P=2 M=3 e P=2 M=2).

3.2. Dataset Heart Disease

Heart Diesase, rispetto a Iris, presenta più attributi e, nonostatnte ciò, l'albero ha le stesse dimensioni: gli attributi che non sono stati utilizzati non sono seriviti per la classificazione, e quindi ciò significa che si sono presentati casi in cui gli esempi hanno avuto stesso target.

Split	P	M	training precision	test precision
76/299	3	0	0.63333	0.3
76/299	2	0	0.9	0.36667
76/299	1	0	0.43333	0.4
76/299	3	3	0.3333	0.3
76/299	3	2	0.36667	0.33333
76/299	2	2	0.6333	0.43333

Anche in questo caso, ridurre la profondità dell'albero di una unità ne aumenta la precisione.

3.3. Dataset Wine

Il Wine dataset presenta anch'esso molti attributi che non vengono utilizzati come nel caso precedente, e i sotto-alberi hanno solo due foglie.

Split	P	M	training precision	test precision
34/178	3	0	0.96667	0.16667
34/178	2	0	1.0	0.2
34/178	1	0	0.36667	0.3333

Siccome i sottoalberi hanno solo due foglie, non avrebbe senso impostare M in quanto sarebbe come impostare P a 2. Come per il dataset precedente, anche qui non sono stati usati tutti gli attributi.

4. Conclusioni

In generale, apportare modifiche alla creazione dell'albero limitando la cardinalità dell'insieme di esempi o la profondità dell'albero, ha dato due risultati differenti:

- Ridurre la profondità dell'albero, anche se solo di 1 unità, aumenta la precisione dell'albero. Ciò è dovuto soprattutto al fatto che, più è profondo l'albero e più si ha noise nel dataset, e quindi si può avere un caso di overfitting. L'overfitting, in generale, si presenta quando la quantità di attributi aumenta nel dataset: si può notare dai risultati di Wine e Iris.

- D'altra parte, cambiare la minima cardinalità porta solo a un peggioramento della precisione dell'albero. Siccome in tutti i dataset si ha uno split su pochi esempi, ciò può significare che quelli usati per creare i nodi rappresentano noise.