Alberi di decisione con valori mancanti

Mochalova Elizaveta

1 Introduzione

In questo programma è stata implementata una delle strategia descritta da Mitchell per la gestione degli esempi di training in caso di mancanza di alcuni valori degli attributi. Ovvero si assegna una probabilità a ogni valore possibile in base alla frequenza osservata dei valori e per riempire i valori mancanti si usa questa probabilità invece di assegnare semplicemente il valore più comune. L'algoritmo è stato applicato a 3 dataset dell'UCI Machine Learning Repository imitando i valori mancanti con una rimozione casuale ed uniforme.

2 Implementazione

2.1 Gestione di dati mancanti

Si assegna una probabilità a ciascun valore di un attributo categorico. Un'attributo si dice categorico se i suoi valori corrispondono a categorie discrete.

2.2 Applicazione dell'algoritmo

Utilizziamo i seguenti tre Data Sets:

- *Wine* [For91] - ci sono 13 attributi per ogni istanza ed un target class.
- *Glass Identification* [Ger87] - ci sono 10 attributi + target class.
- *Iris* [Fis] - ci sono 4 attributi + target class.

2.3 Rimozione degli attributi

Si rimuovono (casualmente ed uniformenente, con probabilità p) alcuni attributi dal dataset. Per fare ciò si genera un valore random e con probabilità p si decide se rimuovere o meno il valore di quell'attributo attraversando tutto il DataFrame.

2.4 Trattamento dei valori mancanti

Durante la costruzione dell'albero nella funzione get_best_split scorrendo attraverso i features si controllano se ci sono dei valori NaN. Nel caso in cui ci sia un tale valore si calcola la probabilità per ciascun altro valore di quella feature e in base a questa probabilità si calcola entropia e information gain. Questa probabilità inoltre viene salvata come uno degli attributi dei nodi foglia, in questo modo si tiene conto della possibilità che ci sia più di un valore mancante moltiplicando le probabilità.

2.5 Calcolo di accuratezze al variare di p

Si riportano le accuratezze sul test set ottenute al variare di p, per p = 0, 0.1, 0.2, 0.5 utilizzando la funzione "metrics.accuracy_score(y_test, y_pred)".

3 Risultati

Probability	Glass (0.926)	Wine (0.933)	Iris (0.868)
0.0	0.926	0.933	0.868
0.1	0.926	0.911	0.868
0.2	0.926	0.956	0.868
0.5	0.889	0.733	0.842

4 Conclusione

Come possiamo notare dalla tabella dell'accuratezza c'è un calo di prestazioni quanto la probabilità di rimozione del valore sia 0.5, infatti inizialmente all'aumentare di p l'accuratezza rimane all'incirca costante. Si può inoltre notare che nel Data Set Iris le prestazioni sono in generale peggiori, ciò potrebbe essere causato dal numero delle classi per questo DataSet.

References

[Fis] R.A. Fisher.

[For91] PARVUS Forina M. et al. 1991.

[Ger87] B. German. 1987.