INTELLIGENZA ARTIFICIALE

GELLI FRANCESCO (N.M. 4921585)

ELABORATO ASSEGNATO PER L'ESAME FINALE:

IMPLEMENTAZIONE ID3 CON DATI MANCANTI

Il progetto è stato svolto in linguaggio Java, usando l'ambiente di sviluppo Eclipse.

-Strutture dati:

I file forniti contententi i dataSet sono in formato ARFF, usato dal software Weka, pertanto ho usato le classi fornite da tale software per rappresentare le entità coinvolte nel progetto, come Data Set, Attributi, ecc. Nel dettaglio:

- Attribute: rappresenta un attributo di un Data Set. Può essere numerico o categorico, nel secondo caso è possibile ottenere il dominio come un oggetto di tipo java.util.Enumeration.

- Instances: rappresenta un intero Data Set, contenente una serie di istanze, attributi e metodi per accedere alla lunghezza del Data Set, agli specifici attributi e istanze, per randomizzare queste ultime, ecc.

- Instance: una singola istanza, contenente un campo per ogni attributo. Fornisce i metodi per accedere ai singoli campi, rappresentati internamente in floating-point (nel caso l'attributo sia categorico, è rappresentato l'indice del corrispondente valore), ed è possibile ottenere tali valori sia come stringhe sia come double.

- DecisionTree: rappresenta l'albero di decisione. Ogni oggetto DecisionTree corrisponde esattamente ad un nodo dell'albero e contiene una lista di nodi figli. L'intero albero dunque, è rappresentato dalla radice. Ogni nodo ha un'etichetta (una stringa) "attribute" che rappresenta l'attributo che divide il Data Set in quel livello ed è una stringa null se il nodo è un nodo foglia. In tal caso sarà diversa da null l'etichetta "result" che indica la classificazione. Gli archi non sono rappresentati esplicitamente, ogni nodo n (tranne la radice) ha un'etichetta "edge" che rappresenta il valore del dominio dell'attributo del nodo padre che si dirama in n.

-Struttura Programma:

Il programma è strutturato in tre package e in una cartella "data" contenente i Data Set scelti. Il package "application" contiene l'algoritmo e le classi che rappresentano le varie funzionalità, quali operazione sul Data Set ("ManagerDataSet"), calcolo dell'Information Gain ("ManagerInfoGain") e la classificazione. "simulation" contiene il main, è così denominato poiché simula una qualche interfaccia con l'utente. Il package "tree" contiene soltanto la classe DecisionTree. Per il display dell'albero sono state usate le funzionalità offerte da javax.swing.tree.

-Algoritmo:

Prima di passare come argomento il Data Set all'algoritmo, questo viene discretizzato, ovvero gli attributi di tipo numerico vengono convertiti in attributi categorici. Questo avviene mediante un oggetto weka.filters.unsupervised.attribute.Discretize. Viene inoltre fatta la divisione tra Training e Testing Set. Questo è compito della procedura "split", che randomizza gli esempi e poi divide, riservando il 70% delle istanze per il training e il 30% per il testing. Randomizzare è un'operazione necessaria in quanto l'ordine iniziale dei Data Set è tutt'altro che casuale, e dividere senza averli randomizzati porterebbe a risultati pessimi. La procedura decisionTreeLearner contiene l'algoritmo. Non ha bisogno di ricevere il Set di attributi, poiché questi sono ottenibili dal Data Set stesso, viene passata invece la stringa ("assegnation") che contiene il valore assegnato all'attributo del nodo padre, serve a settare il campo "edge" del nuovo nodo che verrà creato. La procedura "majorityElement" restituisce il valore più ricorrente per un dato attributo, e "sameClass" restituisce True quando tutti gli esempi sono classificati allo stesso modo. In ogni ricorsione, si seleziona un attributo per dividere il Data Set in un certo livello dell'albero, calcolando per ognuno di questi il corrispondente valore di Information Gain e scegliendo quello con il valore maggiore. Dopodiché si converte il suo dominio in un enumeration e per ogni valore che può assumere si richiama la procedura passandogli il Data Set ristretto. Questo viene calcolato con un oggetto di tipo weka.filters.unsupervised.attribute.Remove, che copia in un nuovo Data Set ("v_dataSet") gli esempi da trasmettere al sottoalbero. La ricorsione termina in tre casi: o quando tutti gli esempi rimasti sono classificati allo stesso modo, o quando tutti gli attributi sono stati testati, ma rimangono ancora alcune instanze, oppure quando non ci sono più esempi. Nel primo caso la foglia dell'albero sarà ovviamente il valore della classe rimasto, nel secondo sarà la classe più ricorrente negli esempi, e nel terzo caso si sceglie la classe più ricorrente negli esempi del Data Set del nodo padre.

-Gestione Dati Mancanti

L'approccio affrontato per gestire la presenza di dati mancanti non prevede di rimpiazzati prima di eseguire l'algoritmo (cosa possibile grazie ad alcuni filtri forniti da Weka), ma di lasciarli nulli e gestire la loro presenza opportunamente durante l'algoritmo. Precisamente un dato mancante crea dei problemi in due momenti: nel calcolo dell'information gain di un attributo e nella classificazione di un'istanza. Infatti se una cella del Data Set non contiene alcun dato, l'istanza corrispondente non verrà contata, né per il calcolo del gain (e dell'entropia), né per l'errore. Nel dettaglio, la funzione "occurrence" che conta quante volte ricorre un valore per un attributo darebbe un valore inferiore di quello aspettato.
Il problema si risolve dunque modificando opportunamente la funzione occurrence. Tale procedura controlla, per ogni istanza che scorre, se il campo corrispondente all'attributo desiderato è mancante. In caso affermativo, invece che il valore della cella (che è null, essendo il dato mancante), si considera il valore più ricorrente in tutto il Data Set corrispondente a quel nodo dell'albero per tale attributo.

Il metodo majority_element viene invocato dalla procedura occurrence, ma siccome anche esso ha bisogno di contare delle occorrenze, in tal caso escludendo i dati mancanti, egli invocherà la procedura "occurrence_without_miss_val", che esclude tali dati dal conteggio.
Allo stesso modo avviene durante la classificazione: nel testare un'istanza si deve confrontare un dato con uno dei valori dei rami dell'albero di decisione, se uno di questi dati è mancante si sceglie per il confronto sempre il valore più ricorrente (in tutto il Data Set).

-Risultati Sperimentali

La necessità di randomizzare il Data Set rende l'errore della classificazione molto variabile. Riporterò dunque in una tabella più esperimenti per ogni Data Set testato. I valori sono in percentuale.(*)

AUTOS	20.97	22.58	32.26	30.65	12.9	24.19	17.74
VOTE	4.58	3.82|0.02	3.05|0.01	6.87|0.01	4.58|0.01	6.11|0.01	5.34|0.01
LABOR	16.67	22.22|0.03	33.33	27.78|0.03	38.89	22.22|0.05	27.68
BREAST-CANCER	37.21|0.01	27.91|0.02	31.4|0.02	32.56|0.03	30.23|0.02	31.4|0.03	38.37|0.01

(*): in ogni cella il primo valore indica l'errore sul Testing Set, il secondo sul Training Set. Se quest'ultimo non compare è uguale a 0.00%.

Tutti questi Data Set presentano dati mancanti.
Sia LABOR che AUTOS portano ad alberi di decisione molto "bassi", sono presenti delle foglie già al primo livello. Questo è dovuto al fatto che l'attributo con l'Information Gain maggiore tra tutti, che sta alla radice, divide il Data Set in modo che ad alcuni sottoalberi venga passato un insieme di esempi veramente ristretto. In particolare LABOR è composto da un numero di esempi molto esiguo rispetto agli altri (solo 57) e rispetto al numero di attributi, che sono molti (16). I risultati corrispondenti, dunque, sono molto aleatori, risentono moltissimo di come avviene la divisione in training e testing set.
VOTE, invece, contiene una grande quantità di esempi (ben 435) e 16 attributi, l'errore misurato è dunque meno flessibile.