Esercitazione 1 – Classi, Aggregazione e Composizione

Si definiscano le classi Java necessarie modellare l'algoritmo di induzione di alberi di decisione in modo da rappresentare gli attributi (classi `Attribute`, `ContinuousAttribute`, `DiscreteAttribute`) e il training set (classe `Data`)

Le visibilità devono essere definite dallo Studente

Classe astratta `Attribute`

La classe modella un generico attributo **discreto** o **continuo**.

Attributi

`String name`: nome simbolico dell'attributo

`int index`: identificativo numerico dell'attributo

Metodi

`Attribute(String name, int index)`

Input: Nome simbolico dell'attributo e identificativo numerico dell'attributo

Output: Nessuno

Comportamento: E' il costruttore di classe. Inizializza i valori dei membri `name`, `index`

`String getName()`

Input: Nessuno

Output: Nome simbolico dell'attributo (di tipo `String`)

Comportamento: Restituisce il valore nel membro `name`;

`int getIndex()`

Input: Nessuno

Output: identificativo numerico dell'attributo

Comportamento: Restituisce il valore nel membro `index`;

Classe `DiscreteAttribute`

Estende la classe `Attribute` e rappresenta un attributo discreto

Attributi

`String values[]`: Array di oggetti String, uno per ciascun valore discreto che l'attributo può assumere.

Metodi

`DiscreteAttribute(String name, int index, String values[])`

Input: valori per nome simbolico dell'attributo, identificativo numerico dell'attributo e valori discreti

Output : Nessuno

Comportamento: Invoca il costruttore della super-classe e avvalora l'array `values[]` con i valori discreti in input.

`int getNumberOfDistinctValues()`

Input: Nessuno

Output : numero di valori discreti dell'attributo

Comportamento: Restituisce la cardinalità dell'array `values[]`

`String getValue(int i)`

Input: indice di un solo valore discreto rispetto all'array `values[]`

Output: valore discreto con indice il parametro di input

Comportamento: Restituisce il valore dell'elemento i dell'array `values[]`

Classe `ContinuousAttribute`

Estende la classe `Attribute` e rappresenta un attributo continuo

Attributi

Metodi

`public CntinuousAttribute(String name, int index)`

Input: valori per nome simbolico dell'attributo, identificativo numerico dell'attributo

Output : Nessuno

Comportamento: Invoca il costruttore della super-classe

Classe Data

Modella l'insieme di esempi di training

Attributi

`Object data [][]`: Matrice nXm di tipo Object che contiene il training set. Il training set è organizzato come (numero di esempi) X (numero di attributi)

`int numberOfExamples`: Cardinalità del training set

`Attribute explanatorySet[]`: Array di oggetti di tipo Attribute per rappresentare gli attributi indipendenti. In questa fase del progetto gli attributi indipendenti sono discreti

`ContinuousAttribute classAttribute`: Oggetto per modellare l'attributo di classe (target attribute). L'attributo di classe è un attributo numerico.

Metodi

`Data(String fileName)`

Input: Nome del file contenente i dati

Output: Nessuno

Comportamento: E' il costruttore di classe. Esegue i seguenti compiti:

1. Avvalora l'array `explanatorySet[]`. Per il dataset servo.dat, istanzia 4 oggetti di tipo `DiscreteAttribute`, uno per ogni attributo indipendente(`motor`, `screw`, `pain`, `vgain`). Ogni attributo indipendente è creato associando ad esso l'array dei valori discreti che esso può assumere. Ad esempio, all'attributo `motor` saranno associati i valori `A`, `B`, `C`, `D`, `E`
2. Avvalora `classAttribute` istanziando un oggetto di tipo `ContinuousAttribute`.
3. Avvalora il numero di esempi (`numberOfExamples=167`).
4. Popola la matrice `data [][]` con gli esempi di training del caso “servo” (167 esempi e 5 attributi, 4 sono indipendenti, 1 è dipentente);

Fornito dal docente

`int getNumberOfExamples()`

Input: Nessuno

Output: Cardinalità dell'insieme di esempi

Comportamento: Restituisce il valore del membro `numberOfExamples`

`int getNumberOfExplanatoryAttributes()`

Input: Nessuno

Output: Cardinalità dell'insieme degli attributi indipendenti

Comportamento: Restituisce la lunghezza dell'array `explanatorySet[]`

`public Double getClassValue(int exampleIndex)`

Input: indice di riga per la matrice `data[][]` per uno specifico esempio

Output: valore dell'attributo di classe per l'esempio indicizzato in input

Comportamento: Restituisce il valore dell'attributo di classe per l'esempio `exampleIndex`

`Object getExplanatoryValue(int exampleIndex, int attributeIndex)`

Input: indice di riga per la matrice `data [][]` per uno specifico esempio

Output: Object associato all'attributo indipendente per l'esempio indicizzato in input

Comportamento: Restituisce il valore dell' attributo indicizzato da `attributeIndex` per l'esempio `exampleIndex`

`Attribute getExplanatoryAttribute(int index)`

Input: indice nell'array `explanatorySet[]` per uno specifico attributo indipendente

Output: oggetto `Attribute` indicizzato da `index`

Comportamento: Restituisce l'attributo indicizzato da `index` in `explanatorySet[]`

`ContinuousAttribute getClassAttribute()`

Input: Nessuno

Output: Oggetto `ContinuousAttribute` associato al membro `classAttribute`;

Comportamento: restituisce l'oggetto corrispondente all'attributo di classe

`public String toString()`

Input: Nessuno

Output: Nessuno

Comportamento: legge i valori di tutti gli attributi per ogni esempio da `data [][]` e li concatena in un oggetto String che restituisce come risultato finale in forma di sequenze di testi.

Fornito dal docente

`void sort(Attribute attribute, int beginExampleIndex, int endExampleIndex)`

Input: Attributo i cui valori devono essere ordinati.

Output: Nessuno

Comportamento: Ordina il sottoinsieme di esempi compresi nell'intervallo `[inf, sup]` in `data[][]` rispetto allo specifico attributo `attribute`. Usa l'Algoritmo quicksort per l'ordinamento di un array di interi usando come relazione d'ordine totale "<=". L'array, in questo caso, è dato dai valori assunti dall'attributo passato in input. Vengono richiamati i metodi: `private void quicksort(Attribute attribute, int inf, int sup); private int partition(DiscreteAttribute attribute, int inf, int sup)`

e `private void swap(int i, int j)`

`public static void main(String args[])`

Comportamento: Consente il test delle classi implementate, in particolare permette la stampa degli esempi ordinati per valori di attributo

Fornito dal docente