Esercitazione 1 – Classi, Aggregazione e Composizione

Si definiscano le classi Java necessarie per modellare l'algoritmo di regressione KNN in modo da rappresentare gli attributi (classi `Attribute`, `ContinuousAttribute`, `DiscreteAttribute`), il training set (classi `Example` e `Data`) e l'algoritmo di data mining(classe `KNN`)

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)`

È il costruttore di classe. Inizializza i valori dei membri `name`, `index`

`String getName()`

Restituisce il valore nel membro `name`;

`int getIndex()`

Restituisce il valore nel membro `index`;

Classe `DiscreteAttribute`

Estende la classe `Attribute` e rappresenta un attributo discreto

Attributi

Metodi

`DiscreteAttribute(String name, int index)`

Invoca il costruttore della super-classe

Classe `ContinuousAttribute`

Estende la classe `Attribute` e rappresenta un attributo continuo

Attributi

Metodi

`public CntinuousAttribute(String name, int index)`

Invoca il costruttore della super-classe

Classe `Example`

Modella i valori degli attributi indipendenti di un esempio

Attributi

`Object example []`: Array di Object che continue un valore per ciascun attributo indipendente

Metodi

`Example(int size)`

costruisce l'array `example` come array di dimensione `size`

`void set(Object o, int index)`

memorizza `o` nella posizione `index` di `example`

`Object get(int index)`

restituisce in valore memorizzato nella posizione `index` di `example`

`void swap(Example e)`

scambia i valori contenuti nel campo `example` dell' oggetto corrente con i valori contenuti nel campo `example` del parametro `e`

`double distance(Example e)`

calcola e restituisce la distanza di **Hamming** calcolata tra l'istanza di `Example` passata come parametro e quella corrente

Classe Data

Modella il training set

Attributi

`Example data[];`
Array di oggetti istanza della classe `Example`

`Double target[];`
Array di valori della variabile target, un valore per ogni esempio (istanza di `Example`) memorizzato in data

`int numberOfExamples;`
numero di esempi memorizzato in `data`

`Attribute explanatorySet[];`
array delle variabili indipendenti che definiscono lo schema degli oggetti istanza di `Example`

`ContinuousAttribute classAttribute;`

attributo target

Metodi

`Data(String fileName)`

È il costruttore di classe.

Fornito dal docente

`int getNumberOfExplanatoryAttributes()`

Restituisce la lunghezza dell'array *explanatorySet[]*

```
private  int partition(double key[], int inf, int sup)
private void quicksort(double key[], int inf, int sup)
```

fornite dal docente per ordinare gli elementi di `data`, `target` e `key` in accordo ai valori contenuti in `key`

`double avgClosest(Example e, int k)`

1) Avvalora `key` con le distanze calcolate tra ciascuna istanza di `Example` memorizzata in **data** ed `e` (usare il metodo `distance` di Example)
2) ordina `data`, `target` e `key` in accordo ai valori contenuti in `key` (usare `quicksort`)
3) identifica gli esempi di `data` che sono associati alle `k` distanze più piccole in `key`

4) calcola e restituisce la media dei valori memorizzati in `target` in corrispondenza degli esempi isolati al punto 3

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

Consente il test delle classi implementate, in particolare permette la stampa del training set

Fornito dal docente

Classe KNN

Modella il miner

Attributi

```
Data data;
```
Modella il training set

Metodi

```
KNN(Data trainingSet)
```

Avvalora il training set

```
Double predict(Example e, int k)
```

Predice il valore target dell'esempio passato come parametro (fare uso di *avgClosest*)

Classe MainTest

Fornite dal docente