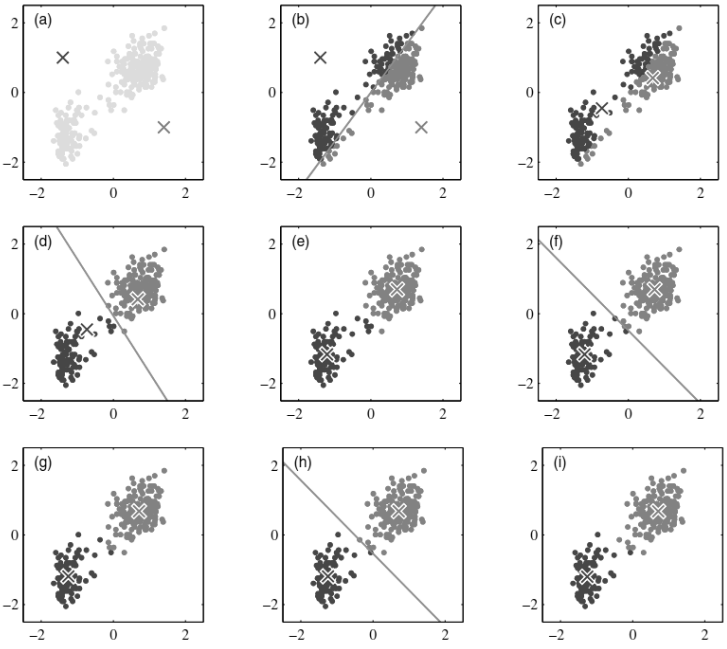
Università degli studi di Catania

Corso di Laurea in Fisica - Primo livello - A.A. 2023-2024
Esame di informatica - 9 settembre 2024
Prof. Marco Russo

Un metodo noto in letteratura per partizionare dei dati in un predeterminato numero di cluster è denominato k-means. Secondo questo metodo avendo a disposizione un insieme di punti n_d in uno spazio n-dimensionale, si possono facilmente individuare un numero predefinito n_c di clusters che partizionano "equamente" i punti originari secondo un operatore di distanza predefinito. A tal uopo occorre iterare in maniera alternata una fase denominata NNC seguita da una denominata CC. L'errore di partizionamento diminuisce in maniera monotonica. Le partizioni sono identificate dai loro baricentri che sono le incognite del problema. Quindi l'algoritmo parte da n_c baricentri casuali e poi inizia le fasi di NNC-CC. Nella fase di NNC si partizionano i punti in base alla vicinanza ai baricentri (centroidi) attuali. Nella CC si "scartano" i centroidi attuali che vengono sostituiti dai centroidi delle partizioni della fase precedente. L'errore associato ad ogni partizione consiste nella sommatoria delle distanza di tutti i punti della partizione dal centroide corrispondente.

Occorre scrivere un programma in C che esegua il k-means nello spazio bidimensionale con distanza di tipo euclideo. Questo programma deve chiedere da tastiera sia n_d che n_c. Deve poi creare 4 array. I primi due saranno gli array dei punti da partizionare e il secondo quello dei baricentri. Gli altri due saranno di parimenti lunghezza e conterranno il primo l'indicazione della partizione di appartenenza del punto corrispondente mentre il secondo conterrà il numero di punti appartenente ad ogni partizione. Occorrerà scrivere anche la funzione **gv** che data una stringa la stampa su video e successivamente acquisisce un valore che verrà ritornato come float. Poi occorre la funzione **dot_casuali** che ha in ingresso un array quasi-statico e lo inizializza con valori casuali in $[0, 1]$. Di seguito abbiamo la funzione **dp** che dati due puntatori a due punti ne torna la distanza euclidea. Un'altra funzione sarà **stampa_punti** che dato un'array di punti lo stampa su video. Le ultime 2 funzioni sono **nnc** e **cc**. Entrmbe hanno in input i 4 array descitti in precedenza. La prima effettua la partizione aggiornando correttamente l'array delle appartenenze e del numero di punti per partizione. La seconda esegue la baricentratura aggiornando i baricentri attuali. Il main deve iterare per 20 volte la NNC seguita dalla CC stampando di volta in volta l'errore totale. Alla fine il programma stampa i centrodi definitivi.

Gli eventuali array inutili abbasseranno sostanzialmente il voto finale! Come di consueto è vietato l'uso di array statici tranne che per le eventuali stringhe. Per acquisire un punteggio in tabella è obbligatorio aver svolto positivamente tutti i punti precedenti. Le eventuali istruzioni scanf e derivate possono acquisire solo stringhe.



Ecco un esempio di output:

```
Num. punti: 1000000
Num. centroidi: 4
It n.0 err=271382.78
It n.1 err=197994.53
It n.2 err=193263.05
It n.3 err=191993.36
It n.4 err=191577.98
It n.5 err=191419.98
It n.6 err=191351.23
It n.7 err=191332.53
It n.8 err=191322.80
It n.9 err=191315.38
It n.10 err=191314.55
It n.11 err=191316.77
It n.12 err=191315.48
It n.13 err=191314.61
It n.14 err=191313.89
It n.15 err=191315.06
It n.16 err=191316.34
It n.17 err=191315.58
It n.18 err=191315.58
It n.19 err=191315.16
Centroidi
(0.75,0.75)
(0.25,0.75)
(0.75,0.25)
(0.25,0.25)
```

Valutazione del compito.

2 punti	Per la Definizione di struttura adeguata per i punti
3 punti	Per la Creazione dei 4 array
2 punti	Per la funzione **gv**
5 punti	Per la funzione **dot_casuali**
3 punti	Per la funzione **dp**
5 punti	Per la funzione **stampa_punti**
5 punti	Per la funzione **nnc**
5 punti	Per la funzione **cc**
5 punti	Per il completamento del compito