K23γ: Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα
Χειμερινό εξάμηνο 2016-17

2η Προγραμματιστική Εργασία
Υλοποίηση του αλγορίθμου συσταδοποίησης K-medoids στη γλώσσα C/C++

Η άσκηση πρέπει να υλοποιηθεί σε σύστημα Linux και να υποβληθεί στις Εργασίες του e-class το αργότερο την Τρίτη 29/11 στις 23.59.

Περιγραφή της άσκησης
Θα υλοποιήσετε τον αλγόριθμο K-medoids για τη συσταδοποίηση δεδομένων που ανήκουν σε σύνολο το οποίο αποτελεί μετρικό χώρο, χρησιμοποιώντας τους 8 συνδυασμούς από τις εξής παραλλαγές:

Initialization
1. K-medoids++
2. Concentrate (Park-Jun)

Assignment
1. PAM assignment (simplest approach)
2. Assignment by LSH/DBH: Reverse Approach

Update
1. Update a la Lloyd's
2. CLARANS Update

Επίσης θα υλοποιήσετε τον αλγόριθμο **CLARA.** Η συσταδοποίηση που παράγει κάθε εκδοχή του αλγορίθμου θα αξιολογηθεί βάσει του δείκτη **Silhouette.**

Το πρόγραμμα θα υποστηρίζει, κατ' ελάχιστο, αντικείμενα που ανήκουν: α) στον d-διάστατο πραγματικό διανυσματικό χώρο βάσει τόσο α1) της ευκλείδειας μετρικής όσο και α2) της απόστασης cosine, β) στον χώρο Hamming και γ) σε μετρικό χώρο όπου η απόσταση ορίζεται μέσω πίνακα αποστάσεων. Ο σχεδιασμός του κώδικα θα πρέπει να επιτρέπει την εύκολη επέκτασή του σε διαφορετικούς χώρους.

ΕΙΣΟΔΟΣ

1) Ένα αρχείο κειμένου `input.dat` διαχωρισμένο με στηλοθέτες (tab-separated), το οποίο θα έχει την ακόλουθη γραμμογράφηση:

```
@metric_space vector
@metric {euclidean, manhattan, cosine} //default: euclidean else define
item_id1    X11     X12     ...     X1d

   .         .       .      ...      .

item_idN    XN1     XN2     ...     XNd
```

όπου **Xij** οι συντεταγμένες double του διανύσματος που αναπαριστά το αντικείμενο i όταν τα αντικείμενα «ζουν» στο d-διάστατο Ευκλείδειο χώρο, **ή**

```
@metric_space hamming
 item_id1       B1

    ...         ...

 item_idN       BN
```

όπου **Bi** η δυαδική συμβολοσειρά μήκους <= 64 bits που αναπαριστά το αντικείμενο i, όταν τα αντικείμενα «ζουν» στο χώρο Hamming, **ή**

```
@metric_space matrix
@items [item_id1, item_id2, ..., item_idN]
0         d12       d13       .         d1N
0         0         d23       .         d2N
.         .         .         .         .
0         0         0         .         0
```

όπου **dij** η απόσταση του αντικειμένου i από το αντικείμενο j (dij = dji, dii = 0), όταν ο μετρικός χώρος προσδιορίζεται από τον πίνακα αποστάσεων του συνόλου των N αντικειμένων.

2) Ένα αρχείο ρύθμισης παραμέτρων cluster.conf με την ακόλουθη μορφή (γραμμές όπου υπάρχει default τιμή μπορούν να μην δινονται οπότε χρησιμοποιείται η default τιμή):

number_of_clusters: <int> // k
number_of_hash_functions: <int> //default:4
number_of_hash_tables: <int> //default:L=5
clarans_set_fraction: <int> //default: max{0.12 * k(N − k), 250}
clarans_iterations: <int> //default:2

Τα αρχεία `input.dat, cluster.conf` δίνονται μέσω παραμέτρων στη γραμμή εντολών. Η εκτέλεση θα γίνεται μέσω της εντολής:
`$./medoids –d <input file> –c <configuration file> -o <output file>`

ΕΞΟΔΟΣ

Ένα αρχείο κειμένου το οποίο περιλαμβάνει τις συστάδες των δεδομένων που παρήχθησαν από κάθε παραλλαγή του αλγορίθμου, τον χρόνο εκτέλεσης σε κάθε περίπτωση καθώς και τον δείκτη εσωτερικής αξιολόγησης της συσταδοποίησης **Silhouette**. Σε κάθε εκτέλεση του προγράμματος πραγματοποιείται συσταδοποίηση των δεδομένων εισόδου και για τους 9 διαφορετικούς αλγόριθμούς. Το αρχείο εξόδου ακολουθεί υποχρεωτικά το παρακάτω πρότυπο, το οποίο επαναλαμβάνεται για κάθε παραλλαγή και για τον αλγόριθμο CLARA:

```
Algorithm: IxAxUx, x∈{1,2} ή CLARA
CLUSTER-1 {size: <int>, medoid: <item_id>}
.      .      .      .      .      .      .
CLUSTER-k {size: <int>, medoid: <item_id>}
clustering_time: <double> //in seconds
Silhouette: [s1,...,si,...,sk,stotal]
/* si=average s(p) of points in cluster i, stotal=average s(p) of points in dataset */

/* Optionally with command line parameter –complete */
CLUSTER-1 {item_idA, item_idB, ..., item_idC}
.      .      .      .      .      .      .

CLUSTER-k {item_idR, item_idT, ..., item_idZ}
```

Επιπρόσθετες απαιτήσεις

1. Το πρόγραμμα πρέπει να είναι καλά οργανωμένο με χωρισμό των δηλώσεων / ορισμών των συναρτήσεων, των δομών και των τύπων δεδομένων σε λογικές ομάδες που αντιστοιχούν σε ξεχωριστά αρχεία επικεφαλίδων και πηγαίου κώδικα. Η μεταγλώττιση του προγράμματος πρέπει να γίνεται με τη χρήση του εργαλείου make και την ύπαρξη κατάλληλου Makefile. Βαθμολογείται και η ποιότητα του κώδικα (π.χ. αποφυγή memory leaks).

2. Στην υλοποίηση μπορείτε να χρησιμοποιείτε συναρτήσεις από την πρότυπη βιβλιοθήκη της C/C++. Αντίθετα δεν επιτρέπεται η χρήση βιβλιοθηκών για τις απαραίτητες δομές δεδομένων (συνδεδεμένες λίστες, vectors, hashtables), για τις γεννήτριες τυχαίων αριθμών (εκτός από τις συναρτήσεις που δηλώνονται στο `stdlib.h` της πρότυπης βιβλιοθήκης της C), για τις συναρτήσεις κατακερματισμού, για τις συναρτήσεις υπολογισμού αποστάσεων ούτε για άλλη λειτουργία του προγράμματος.

3. Το παραδοτέο πρέπει να είναι επαρκώς τεκμηριωμένο με πλήρη σχολιασμό του κώδικα και την ύπαρξη αρχείου readme το οποίο περιλαμβάνει κατ' ελάχιστο: α) τίτλο και περιγραφή του προγράμματος, β) κατάλογο των αρχείων κώδικα / επικεφαλίδων και περιγραφή τους, γ) οδηγίες μεταγλώττισης του προγράμματος, δ) οδηγίες χρήσης του προγράμματος και ε) πλήρη στοιχεία των φοιτητών που το ανέπτυξαν.

4. Η υλοποίηση του προγράμματος θα πρέπει να γίνει με τη χρήση συστήματος διαχείρισης εκδόσεων λογισμικού και συνεργασίας (Git ή SVN).

5. Χρήση κατάλληλης βιβλιοθήκης και εκτέλεση ελέγχων μονάδων λογισμικού (unit testing).