Ονομα: Θεοδώρα
Επώνυμο: Αναστασίου
Α.Μ : 1115201400236
Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα : Εργασία 2

Παρατηρήσεις:
1)Το πρόγραμμα έχει υλοποιηθεί σε c++ .

2)Στον φάκελο περιέχεται αρχείο Makefile το οποίο μεταγλωτίζει το πρόγραμμα.

3)Εκτελείται με τις εξής εντολές :
make
./cluster -i input_small_cv -c configuration_file -o output -d euclidean/cosine -complete/η κενο

4)Το έχω τρεξει με valgrind για απαλοιφή errors/leaks

5) Υπάρχει στο φάκελο αρχείο cluster.conf που μου δίνει κάποιες σημαντικές πληροφορίες για την εκτέλεση του προγράμματος

number_of_clusters: <int> //k
number_of_hash_functions: <int> //default:4
number_of_hash_tables: <int> //default:L=5

(για τα default αποτελέσματα απλά αφήνω το <int> όπως είναι και πάραπάνω.)

6) Κατα την εκτέλεση του προγράμματος δημιουργείται αρχείο με τα δεδομένα και τους υπολογισμούς που ζητήθηκαν. (metric,centroid id or coordinates, clustering time, silhouette) και ανάλογα με το – complete εκτυπώνει πέρα απο τα πάραπάνω και τα αντικείμενα που περιέχει κάθε cluster.

7)Το πρόγραμμα μπορεί να τρέξει και τους 12 συνδιασμούς αλλά δίνεται δυνατότητα να τρέξει μόνο ένας δίνοντας στο cluster.conf αριθμό που αντιστοιχεί σε αριθμό συνδιασμού .

Give 0 for : ALL
Give 1 for : Random selection + Lloyds + Kmeans
Give 2 for : Random selection + Lloyds + PAM
Give 3 for : KmeansPlusPlus + Lloyds + Kmeans
Give 4 for: KmeansPlusPlus + Lloyds + PAM
Give 5 for: Random selection + LSH + Kmeans
Give 6 for : Random selection + LSH + PAM
Give 7 for:KmeansPlusPlus + LSH + Kmeans
Give 8 for :KmeansPlusPlus + LSH + PAM
Give 9 for: Random selection + Hypercube + Kmeans
Give 10 for : Random selection + Hypercube + PAM
Give 11 for:KmeansPlusPlus + Hypercube + Kmeans
Give 12 for :KmeansPlusPlus + Hypercube + PAM

Έχω υλοποιήσει τις εξής δομές/αρχεία :

- (απο 1η εργασια διαμορφωμένες κατάλληλα για να δέχονται δεδομένα τύπου double)
DataSet.cpp/DataSet.h , Hash_Euclidean.cpp/Hash_Euclidean.h, Hash_Cosine.cpp/Hash_Cosine.h, HF_Bucket.cpp/HF_Bucket.h, Lsh.cpp/Lsh.h, HyperCube.cpp/Hypercube.h

- Καθώς επίσης και τα εξής εξολοκλήρου για το 2ο κομμάτι της εργασίας:

Cluster.cpp,Cluster.h
Έχω δημιουργήσει μια δομή Cluster η οποία περιέχει ένα αναγνωριστικό αριθμό (id) του κέντρου του, τα αντικείμενα που ανήκουν στο συγκεκριμένο cluster, και τον vector με τις συντεταγμένες του κέντρου του.

Combinations.cpp, Combinations.h,
Περιέχουν μέσα συναρτήσεις που καλούν τις κατάλληλες συναρτήσεις ούτως ώστε να ικανοποιούνται όλοι οι συνδιασμοί αλγορίθμων.

notfun.cpp,notfun.h,
Περιέχουν τις συναρτήσεις για τους αλγορίθμους καθώς και άλλες πιο μικρές βοηθητικές συναρτήσεις για την σωστή λειτουργία των συναρτήσεων των αλγορίθμων.

Project.cpp → ουσιαστικά η main , διαβάζει το αρχείο με τα δεδομένα, φτιάχνει το dataset, διαβάζει τις παραμέτρους απο το αρχείο cluster.conf και τις διαχειρίζεται ανάλογα, καλεί ανάλογα με το τί θα επιλεξει ο χρήστης να εκτελέσει και τις κατάλληλες συναρτήσεις απο το αρχείο Combinations.cpp για τους αντίστοιχους αλγορίθμους.

INITIALIZATION :

- ***Random Selection of k points*** :
Ουσιαστικά απο το dataset παίρνω k τυχαία items και τα ορίζω σαν clusters.

- ***Kmeans++*** :
Ουσιαστικά το 1ο cluster επιλέγεται τυχαία απο το dataset ,τα επόμενα επιλέγονται υπολογίζοντας την απόσταση τους απο το προηγούμενο ούτως ώστε να έχουν περισσότερη πιθανότητα να επιλεγούν σαν κέντρα τα items που βρίσκοντε πιό μακριά απο τα προηγούμενα του.

ASSIGNMENT :

- Lloyds Assignment:
Υπολογίζει εξαντλητικά όλες τις αποστάσεις των αντικείμενων του dataset απο όλα τα κεντρα των cluster που αρχικοποιήθηκαν στο προηγούμενο βήμα, και τα αναθέτει στο πιό κοντινό τους.

-LSH: Δημιουργώ το αντικείμενο LSH οπως και στο 1ο κομμάτι του Project, με δεδομένα τα L,h απο το configuration_file. Δημιουργούνται τα buckets και μπαίνουν μέσα αναλόγως τα items του DataSet. Στην συνεχεια κάθε cluster εκτελεί τον ρόλο του query της προηγούμενης εργασίας, μου επιστρέφεται μετά απο πράξεις και συναρτήσεις απο την προηγούμενη εργασία ένας αριθμός που αντιστοιχεί σε ένα bucket. Στην συνέχεια ανατρέχω τα στοιχεία του συγκεκριμένου bucket και αν είναι μέσα στο Range τότε ελέγχω αν έχει γίνει ξανά assign στην ίδια επανάληψη(rev) αν έχει γίνει ξανά στην ίδια επανάληψη τότε τσεκάρω αν η απόσταση του προηγούμενου κέντρου στο οποίο είχε ανατεθεί είναι μικρότερη απο του νέου που ίσως ανατεθεί τα συγκρίνω και το αντικείμενο ανατίθεται στο κέντρο με την πιο μικρή απόσταση απο το αντικείμενο.

Αν δεν είχε ανατεθεί ποτέ τότε το αντικείμενο απλά ανατίθεται στο συγκεκριμένο κεντρο.
Αφού τελείωσει όλη η διαδικασία αυτή για όλα τα clusters ελέγχω με μετρητές εάν τα αντικέιμενα των buckets έχουν ανατεθεί όλα και σταματάω, αλλιώς κοιτάζω αν έχουν βρεθεί όντως και άνατέθηκαν στοιχεία στην συγκεκριμένη επανάληψη, αν ισχύει το πρώτο τότε σταματάω γιατί δεν έχω άλλα στοιχεία να αναθέσω, αν ισχύει το 2ο επίσης σταματάω και στην συνέχεια τσεκάρω αν υπάρχουν αντικείμενα τα οποία δεν έχουν ανατεθεί σε κάποιο cluster και τα αναθέτω με τον αλγόριθμο Lloyds.

-HyperCube:
Η αρχικοποίηση του HyperCube και η διαδικασία κατανομής των αντικειμένων στα Buckets είναι ακριβώς η ίδια με της προηγούμενης εργασίας. Στην συνέχεια σαν “query” μπαίνει το διάνυσμα του κέντρου του κάθε Cluster και στην συνέχεια επιλέγεται ένα Bucket στο οποίο θα άνηκε αυτό το διάνυσμα. Στην συνέχεια βάση του Bucket αυτού υπολογίζονται και οι γείτονές του όπως και στην προηγούμενη εργασία , και ανάλογα με τα probes-M . Δημιουργείται μια λίστα με αντικείμενα που θα πρέπει να κατανεμηθούν στα Clusters, και απο εκεί και πέρα ακολουθείται η διαδικασία που ακολουθήσα και στην υλοποίηση του LSH-Clustering. (περιγράφεται πιο πάνω)

UPDATE:

-Kmeans:
Ψάχνει να βρεί το επόμενο κέντρο ουσιαστικά κάνοντας την εξής διαδικασία.
Για κάθε στοιχείο μέσα στο cluster παίρνει κατα στήλη τις τιμές των συντεταγμένων του και τις προσθέτει με τις αντίστοιχες των επόμενων στοιχείων του ίδιου cluster(απλά έχω ένα vector που αρχικά μηδενίζει τα όσα και τα αντικείμενα στο cluster ,στοιχεία του, και στην συνέχεια απλά προσθέτω κατάλληλα τις συντεταγμένες στην αντίστοιχη θέση του vector μου).
Στο τέλος απλά διαιρεί / πλήθος των αντικειμένων του cluster, δηλαδή στην δική μου εκτέλεση παίρνω τον vector που έφτιαξα προηγουμένος και διαιρώ όλα του τα στοιχεία με το πλήθος των στοιχείων στο συγκεκριμένο cluster.
Όταν γίνει αυτή η διαδικασία έχω το καινούριο μου κεντρο.
Το τοποθετώ σε έναν προσωρινό vector απο κεντρα, όταν έχω και τα k (νεα /πιθανα)κεντρα μου, τα συγκρίνω με τα προηγούμενα κεντρα και αν δεν έχουν τις ίδιες συντεταγμένες τότε τα ορίζω τελικά σαν νέα κέντρα και καλώ ξανά την συνάρτηση Assignment Lloyds ,αλλιώς τερματίζει η εκτέλεση του Update..

-PAM Improved like Lloyd’s:
Ψάχνει το medoid (δηλαδη το σημείο που είναι πιο κοντά σε όλα τα άλλα σημεία μέσα στο cluster . Το medoid υπολογίζεται για κάθε cluster, για κάθε αντικείμενο i, για κάθε αντικείμενο j, προσθέτει την απόσταση του j απο το i, όταν τελειώνουν όλοι οι υπολογισμοί για το αντικείμενο i (υποψήφιο medoid), η απόσταση αποθηκεύεται σε ένα vector<double>, όταν τελειώσουν οι υπολογισμοί για όλα τα αντικείμενα i, τότε ψάχνω να βρώ την μικρότερη τιμή απόστασης στον vector<int> μου , πέρνει την θέση και το id του αντικειμένου στο οποίο αντιστοιχεί και το ορίζω σαν το νέο κέντρο μου. Το τοποθετώ σε έναν προσωρινό vector απο κεντρα, όταν έχω και τα k (νεα /πιθανα)κεντρα μου, τα συγκρίνω με τα προηγούμενα κεντρα και αν δεν έχουν τις ίδιες συντεταγμένες τότε τα ορίζω τελικά σαν νέα κέντρα και καλώ ξανά την συνάρτηση Assignment Lloyds ,αλλιώς τερματίζει η εκτέλεση του Update.

Silhouette :
Φτιάχτηκε με βάση τον τύπο απο τις διαφάνειες σελ.46
a(i) η μέση αποσταση απο τα αντικείμενα στο ίδιο cluster
b(i) η μέση απόσταση απο τα αντικείμενα στο δεύτερο καλύτερο cluster που μπορούσε να μπεί το αντικείμενο αυτο s(i)=α(ι)-β(ι)/μαξ

ΣΥΓΚΡΙΣΕΙΣ ΑΛΓΟΡΙΘΜΩΝ :

Random – Lloyds – Update_kmeans:

K	Total Silhouette	Update Times	Duration	Metric
5		20	6.15078	Euclidean
10	0,05 0,07 0,14 0,10 0.13	~19	11.145	Euclidean
50	0.18(too slow)			Euclidean
300	0.30355	25	473.807sec	Euclidean
10	0.0820556	23	23.0013	Cosine

Kmeans++ – Lloyds – Update_kmeans:

K	Total Silhouette	Update Times	Duration	Metric
10	0.106659	23	12.7073sec	Eucidean
10	0.0842514	30	30.6259	Cosine
50	0.164819	27	134.057	Cosine

Kmeans ++ – Lloyds – PAM:

K	Total Silhouette	Update Times	Duration	Metric
5	0.3			Euclidean
10	0.2			Euclidean
40	0.319625	3	425.886sec/	Euclidean
50	0.33267			Euclidean
10	0.155616	2	377.832	Euclidean
10	0.053766	1	149.313	Cosine

Random ++ – Lloyds – PAM: **elegxos k g perissotera clusters

K	Total Silhouette	Update Times	Duration	Metric
5	0.131205	1	418.0secs/6.9mins	Euclidean
10	0.333602		757.1secs /12.6mins	Euclidean
10	0.169175	1	370.544	Euclidean
10	0.077015	2	206.288	Cosine
50	0.131809	2	55.3609	Cosine

KmeansPlusPlus + LSH + Kmeans: (2-2-1) input : 7

K	h	L	Total Silhouette	Update Times	Duration	Metric
5	5	1	0.0391374	15	5.09526	Euclidean
5	5	10	0.0515524	22	8.91878	Euclidean
10	5	5	0.0597475	21	13.5772	Euclidean
10	5	5	0.0739368	30	34.726	Cosine
10	5	10	0.0777742	31	35.916	Cosine

Random selection + LSH + PAM: (1-2-2) input :6

K	h	L	Total Silhouette	Update Times	Duration	Metric
10	5	10	0.211219	2	512.374	Euclidean
10	5	5	0.187422	2	342.947	Euclidean
10	5	5	0.0550411	3	716.642	Cosine

RandomSelecton+ LSH + Kmeans: **elegxos k g perissotera clusters

K	h	L	Total Silhouette	Update Times	Duration	Metric
5	4	3	0.214341	17	5.84487	Euclidean
10	5	5	0.074747	18	11.1735	Euclidean
10	5	5	0.0592868	13	8.96796	Euclidean
5	5	10	0.0511302	21	14.0481	Cosine
10	5	5	0.0773396	23	23.8006	Cosine
10	5	10	0.0705823	25	24.9883	Cosine

KmeansPlusPlus + LSH + PAM : (2-2-2) input 8

K	h	L	Total Silhouette	Update Times	Duration	Metric
10	5	1	0.146548	3	513.201	Euclidean
10	5	5	0.0202506	2	340.152	Cosine

Random selection + Hypercube + Kmeans : (1-3-1) input 9

K	h	L	Total Silhouette	Update Times	Duration	Metric
10	5	1	0.10823	11	11.9167	Euclidean
10	5	1	0.238163	30	17.51	Euclidean
10	5	1	0.0833643	29	28.829	Cosine

Random selection + Hypercube + PAM: **elegxos k g perissotera clusters 1_3_2 input :10

K	h	L	Total Silhouette	Update Times	Duration	Metric
10	5	1	0.15868	2	551.305	Euclidean
10	5	1	0.0516475	3	664	Cosine

KmeansPlusPlus + Hypercube + Kmeans:

K	h	L	Total Silhouette	Update Times	Duration	Metric
10	5	1	0.0890022	27	26.4822	Cosine
10	5	1	0,0844679	20	19.716	Cosine
10	5	1	0.127216	15	9.8471	Euclidean
10	5	1	0.132602	18	15.7608	Euclidean

KmeansPlusPlus + Hypercube + PAM : (2_3_2) input:12

K	h	L	Total Silhouette	Update Times	Duration	Metric
10	5	1	0.160078	2	413.149	Euclidean
10	5	1	0.078974	2	310.651	Cosine
10	5	1	0.164296	2	643.832	Euclidean
10	5	1	0.0316129	2	468.96	Cosine

Γενικά συμπεράσματα:

1) Ο Kmeans είναι γρηγορότερος απο τον PAM.
2) Σαν μετρική η ευκλείδια είναι πιο γρήγορη αλλά παρατήρησα ότι η cosine δίνει καλύτερα αποτελέσματα βάση της Silhouette.
3) Όταν είχα σαν μετρική την Cosine χρειάζονταν περισσότερες επαναλήψης Update-Assignment μέχρι να σταθεροποιηθούν τα κέντρα.
4) Η μέθοδος LSH για Assignment δίνει πολύ καλύτερα αποτελέσματα σε σχέση με το απλό Lloyds.
5) Η LSH είναι πιο γρήγορη απο την μέθοδο Assignment Lloyds.
6) Ο HyperCube είχε μεγαλύτρους χρόνους απο το LSH ενω έχει μόνο 1 table για KmeansUpdate (καθυστερήσεις λόγο γειτόνων)
7) Με Cosine στον hypercube έχουμε καλύτερη Silhouete στις ίδιες επαναλήψεις και μερικές φορές και με καλύτερο χρόνο.