ΜΑΥΡΟΦΟΡΑΚΗ ΔΗΜΗΤΡΑ
1115201400104

2η ΕΡΓΑΣΙΑ

Υλοποίηση των αλγορίθμων συσταδοποίησης K-means / K-medoids στη γλώσσα C/C++

Το πρόγραμμά μου υλοποιεί τους αλγορίθμους για την συσταδοποίηση διανυσμάτων με μετρική το συνημίτονο και την ευκλείδια απόσταση.

- **Διάρθρωση αρχείων κώδικα**

cluster.cc : Η main του προγράμματος μου. Σ' αυτό το αρχείο διαβάζω το dataset και το αποθηκεύω σ έναν πίνακα ενώ ταυτόχρονα δημιουργώ και αρχικοποιώ την δομή του υπερκύβου και τους L hashtables. Στη συνέχεια ξεκινάω επαναλήτικα την εκτέλεση των αλγορίθμων. Μετά το τέλος εκτέλεσης της τριάδας των αλγορίθμων (initialization, assignment, update) εκτυπώνω τα αποτελέσματα τους στο αρχείου εξόδου, και συνεχίζω με την επόμενη τριάδα. Τέλος διαγράφω τις δομές που δέσμευσα δυναμικά.

datastructs.h : Header που περιλαμβάνει τις κλάσεις και τις συναρτήσεις που περιλαμβάνει το πρόγραμμα.

datstructs.cc : Υλοποιεί τις συναρτήσεις των κλάσεων DataVector, Euclidean, Cosine και Cluster.

general_functions.cc : Περιλαμβάνει συναρτήσεις γενικής χρήσης που βασικός τους σκοπός είναι να απομονώσει εργασίες και να συμβάλλει στη διατήρηση ενός διαχειρίσιμου μεγέθους του cluster.cc. Τέτοιες συναρτήσεις είναι για παράδειγμα η εκτύπωση των αποτελεσμάτων, η ανάγνωση του configuration file αλλά και του dataset και συναρτήσεις που καθορίζουν ποιός αλγόριθμος initialization, update και assignment θα εκτελεστεί.

algorithms.h: Header με τους ορισμούς των συναρτήσεων συσταδοποίησης

algorithms.cc : Περιλαμβάνει την υλοποίηση των αλγορίθμων συσταδοποίησης.

Makefile: Αρχείο για την μεταγλώττιση του προγράμματος.

cluster.conf: Configuration file της παρακάτω μορφής.

number_of_clusters:7
number_of_hashfunctions:4
number_of_hashtables:5
w:2
probes:8
M:1000
k:5 //για τη διασταση του hypercube

- **Σχόλια για την επιλογή των δομών και των αλγορίθμων**

Η βασική κλάση των σημείων που διαβάζονται από τα αρχεία είναι η DataVector. Ανάλογα με το αν έχει επιλεγεί να χρησιμοποιηθεί ευκλείδια μετρική ή μετρική συνημιτόνου δημιουργούνται δυναμικά objects τυπου Cosine ή Euclidean (κλάσεις παιδιά της DataVector). Ωστόσο όλες οι συναρτήσεις έχουν όρισμα τύπου DataVector * για να είναι generic. Στην κλάση αυτή αποθηκεύονται και το πιο κοντινό και το δεύτερο πιο κοντινό cluster στο οποίο ανήκει κάθε στοιχείο καθώς και οι αποστάσεις από τα κεντροειδή αυτών.

Επίσης, υπάρχει και η κλάση cluster με βασικά στοιχεία το κεντροειδές (τύπου DataVector) αλλά και μία λίστα με τα σημεία που περιλαμβάνει το cluster. Εάν το κεντροειδές δεν ανήκει στο dataset υπάρχει flag που το δηλώνει (centroid_is_external). Επίσης, στη κλάση αποθηκεύεται και η πληροφορία σχετικά με το αν το cluster έχει υποστεί κάποια αλλαγή μετά την επανάληψη κάποιου αλγορίθμου ανάθεσης (modified). Αυτή η πληροφορία βοηθάει στην μείωση του χρόνου εκτέλεσης του pam καθώς ελέγχει μόνο τα clusters τα οποία τροποποιήθηκαν στο πιο πρόσφατο assignment.

Παρατηρήσεις:
Το Number of hashfuctions δίνεται ως παράμετρος στο configuration file Και δεν επιτρέπω στο χρήστη να είναι πάνω από 18 καθώς δεν μπορεί να δεσμευτεί τόσο χώρος (2^18 πίνακας από λίστες) στην περίπτωση δημιουργίας του hypercube με αποτέλεσμα την υπερχείληση. Ο hypercube δεσμεύεται στατικά. Ωσόσο και στην περιπτωση δυναμικής δομής πάλι υπάρχει περιορισμός στις θέσεις καθώς ο σωρός επιτρέπει να δεσμευτεί περισσότερος χώρος από τη στοίβα ωστόσο δεν είναι απεριόριστος ούτε αυτός.

- **Μεταγλώττιση**

make

- **Εκτέλεση**

./cluster -i sample_datasets/Ergasia_2/siftsmall/twitter_dataset_small.csv -c cluster.conf -o output -d euclidean

Με την παραπάνω εντολή εκτελούνται σειριακά όλοι οι συνδυασμοί αλγορίθμων initialization, assignment, update. Λόγω του μεγάλου χρόνου που κάνει για την εκτέλεση όλων των αλγορίθμων , μπορούμε να τρέξουμε και παραμετρικά το πρόγραμμα ως εξής :
./cluster -i sample_datasets/Ergasia_2/siftsmall/twitter_dataset_small.csv -c cluster.conf -o output -d euclidean -initialization 1 -assignment 1 -update 1

FLAGS:
random initialization :1
plus initialization: 2
lloyds assignment: 1
lsh assignment:2
hypercube assignment: 3
lloyds update : 1
pam update: 2

- **Παρατηρήσεις αλγορίθμων (small dataset)**

<u>random initialization</u>

Δημιουργεί τα νέα clusters και τα αρχικοποιεί με centroid ένα στοιχείο του dataset τύπου DataVector *.

<u>kmeans++ initialization</u>

Επιλέγει τα στοιχεία του dataset που θα γίνουν τα κεντροειδή των νεων clusters με βάσει της μεγαλύτερης πιθανότητας. Προκειμένου να αποφύγω τη δέσμευση ξεχωριστής δομής, δεν αποθηκεύω το άθροισμα των τετραγώνων σε πίνακα. Η δομή δεν μου προσφέρει κάτι καθώς οι αποστάσεις θα πρέπει να επαναϋπολογίζονται μετά την επιλογή ενός νέου κεντροειδούς. Γι' αυτό αποθηκεύω σε μία μεταβλητή το άθροισμα επαναληπτικά. Μόλις το άθροισμα ξεπεράσει το νέο νούμερο, η επανάληψη σταματάει και ο αριθμός της επανάληψης δηλώνει ποιό θα είναι το κεντροειδές πού θα επιλεγεί.

Ενδεικτικές εκτελέσεις :

- *I2A1U1 ,number_of_clusters:20 , number_of_hashfunctions:4, number_of_hashtables:5, w:2. probes:8, M:1000*

clustering time: 43.4691 sec
total silhouette : 0.0910306

- *I2A2U1 ,number_of_clusters:20 , number_of_hashfunctions:4, number_of_hashtables:5, w:2. probes:8, M:1000, unassigned points 91, euclidean metric*

Clustering time: 79.23 sec

total silhouette: 0.0612921

- *I2A3U1 ,number_of_clusters:20 , number_of_hashfunctions:4, number_of_hashtables:5, w:2. probes:8, M:1000, euclidean metric*

Clustering time: 35.0244
total silhouette: 0.0147296

Η βελτίωση στην περίπτωση που το initialization συνδυάζεται με το lloyds assignment είναι πιο εμφανής. Στη περίπτωση του lsh και του hypercube assignment δεν υπάρχει σοβαρή αλλαγή στο total silhouette ίσως επειδή ο κατακερματισμός που προσφέρουν από μόνοι τους αυτοί οι αλγόριθμοι συμβάλει στη συγκέντρωση κοντινών στοιχείων στο bucket των κεντροειδών είτε αυτά έχουν αρχικοποιηθεί με random τρόπο, είτε με πιθανοτικό. Τέλος επιβαρύνει χρονικά το πρόγραμμα αυξάνοντας την πολυπλοκότητά του.

<u>lloyds assignment</u>
Ελέγχει εξαντλητικά για κάθε σημείο του πίνακα ποιά είναι η απόσταση από τα clusters και βρίσκει το κοντινότερο cluster και το neighbour cluster και τις αποστάσεις από τα κεντροειδή.

Ενδεικτικές εκτελέσεις :

- *I1A1U1 ,number_of_clusters:20 , number_of_hashfunctions:4, number_of_hashtables:5, w:2. probes:8, M:1000*

clustering time: 45 sec
total silhouette : 0.0824411

clustering time: 27.3195 sec
total silhouette 0.0750941

Όσο περισσότερο χρόνο κάνει ο αλγόριθμος (όσο πιο πολλές επαναλήψεις κάνει βάσει της βελτιωμένης hash function) τόσο πιο ακριβής είναι ο αλγόριθμος.

- *I1A1U1 ,number_of_clusters:100 , number_of_hashfunctions:4, number_of_hashtables:5, w:2. probes:8, M:1000*

clustering time: 206.174 sec
total silhouette: 0.150946

Όσο μεγαλύτερος είναι ο αριθμός των clusters τόσο καλύτερο είναι το total silhouette γιατί τα clusters είναι πιο ομοιογενή.

<u>lsh assignment</u>

Για την περίπτωση του lsh ελέγχω το bucket στο οποίο αντιστοιχεί κάθε κεντροειδές και κάνω assign τα σημεία του στο κεντροειδές. Τα σημεία που μένουν unassigned ενημερώνονται από το lloyds assignment. Η επιλογή των παραμέτρων του lsh (ειδικά το w) παίζει ιδιαίτερο ρόλο στο clustering. Επιθυμώ “κακό” hashing ώστε τα buckets να έχουν όσο το δυνατόν περισσότερα στοιχεία και να φανεί και ξεκάθαρα το αποτέλεσμα του lsh assignment χωρίς να χρειαστεί να γίνουν assign πολλά στοιχεία a la lloyds. **Η ποιότητα του hashing και η ποιότητα του clustering είναι αντιστρόφως ανάλογες.**

Ενδεικτικές εκτελέσεις :

- *I1A2U1 ,number_of_clusters:5 , number_of_hashfunctions:4, number_of_hashtables:5, w:2. probes:8, M:1000, euclidean, unassigned σημεία 444*

Clustering time: 29.2729
Total silhouette : 0.0453825

- *I1A2U1 ,number_of_clusters:20 , number_of_hashfunctions:4, number_of_hashtables:5, w:2. probes:8, M:1000, euclidean, unassigned σημεία 82*

clustering time: 66 sec
total silhouette: 0.0641702

- *I1A2U1 ,number_of_clusters:20 , number_of_hashfunctions:4, number_of_hashtables:5, w:2. probes:8, M:1000, cosine*

clustering time: 50.5361 sec
total silhouette: 0.0320633

- *I1A2U1 ,number_of_clusters:100 , number_of_hashfunctions:4, number_of_hashtables:5, w:2. probes:8, M:1000, euclidean*

clustering time: 290.854 sec
total silhouette: 0.0798281

Σε σχέση με τα αποτελέσματα του lloyds το total silhouette είναι χειρότερο καθώς πλέον η αναζήτηση του κοντινότερου cluster γίνεται προσεγγιστικά και όχι εξαντλητικά. Ωστόσο ο χρόνος δεν βελτιώθηκε ιδιαίτερα. Παρ’ όλο που αποφύγαμε την επανάληψη για το κοντινότερο γείτονα δαπανήθηκε χρόνος για την αναζήτηση του δεύτερου κοντινότερου γείτονα. Αυτό εξηγεί τα χρονικά αποτελέσματα.
Επίσης ο ευκλείδιος Lsh έδωσε καλύτερα αποτελέσματα από τον cosine.

hypercube assignment

Όπως και στον lsh έτσι και στην περίπτωση του υπερκύβου οι παράμετροι καθορίζουν το clustering. Προκειμένου να έχουμε μια αντικειμενική εικόνα του hypercube assignment πρέπει να έχουμε έναν μεγάλο αναλογικά με τη διάσταση του υπερκύβου αριθμό Probes ώστε να γίνονται assigned πιο πολλά στοιχεία έτσι παρά με Lloyds. Παρατήρησα ότι στον υπερκύβο το cosine και η ευκλείδια μετρική δίνουν παρόμοια αποτελέσματα. Προσοχή: ο χρήστης δεν έχει την δυνατότητα να δώσει σαν argument τον αριθμό των clusters να υπερβαίνει το συνολικό μέγεθος του υπερκύβου (array από λίστες στοιχείων).

Ενδεικτικές εκτελέσεις :

- *I1A3U1 ,number_of_clusters:10 , number_of_hashfunctions:4, number_of_hashtables:5, w:2. probes:8, M:1000, euclidean*

Clustering time: 15.4626
Total silhouette: 0.0111482

- *I1A3U1 ,number_of_clusters:10 , number_of_hashfunctions:4, number_of_hashtables:5, w:2. probes:8, M:1000, euclidean*

Clustering time: 8.36688
Total silhouette: 0.0189098

- *I1A3U1 ,number_of_clusters:5 , number_of_hashfunctions:4, number_of_hashtables:5, w:2. probes:8, M:1000, euclidean (run 1)*

Clustering time: 8.88298
Total silhouette: 0.0391191

- *I1A3U1 ,number_of_clusters:5 , number_of_hashfunctions:4, number_of_hashtables:5, w:2. probes:8, M:1000, euclidean (run 2)*

Clustering time: 6.74931
Total silhouette : 0.0205525

- *I1A3U1 ,number_of_clusters:7 , number_of_hashfunctions:4, number_of_hashtables:5, w:2. probes:8, M:1000, cosine*

Clustering time: 14.037
Total silhouette : 0.012099

Lloyd's update

Σε κάθε Lloyd's update δημιουργείται ένα νέο object τύπου DataVector για κάθε cluster, με συντεταγμένες που προκύπτουν από το μέσο όρο των στοιχείων τους. Ο πίνακας με τα clusters ενημερώνεται για τα νέα κέντρα τους.

Ο Lloyd's update βελτίωσε σε ικανοποιητικό βαθμό και τις τρεις μεθόδους assignment , τα αποτελέσματα του οποίου είναι εμφανή στις παραπάνω εκτελέσεις.

PAM update

Στην περίπτωση του PAM επιλέγεται ένα στοιχείο από το dataset για να γίνει το νέο κεντροειδές (δεν δημιουργούμε καινούργιο DataVector).

Ενδεικτικές εκτελέσεις :

- *I1A1U2 ,number_of_clusters:5 , number_of_hashfunctions:4, number_of_hashtables:5, w:2. probes:8, M:1000, euclidean*

Clustering time: 917.209 sec
total silhouette : 0.0429472

- *I1A2U2 ,number_of_clusters:5 , number_of_hashfunctions:4, number_of_hashtables:5, w:2. probes:8, M:1000, euclidean*

Clustering time: 896.621 sec
total silhouette : 0.0421487

Το total silhouette δεν βελτιώνεται αισθητά με τη χρήση του PAM . Επομένως, δεδομένης και της μεγάλης χρονικής πολυπλοκότητας δεν το προτείνω σαν update μέθοδο για το συγκεκριμένο dataset.