Locality Sensitive Hashing , HyperCube and data Clustering

Τατάς Μιχαήλ 11151700161
Φωτιάδης Μιχαήλ 11151700183

Περιεχόμενα

1 Οδηγίες μεταγλώτισσης και χρήσης του προγράμματος

Το πρόγραμμα μεταγλωτίζεται με την εντολή make. Όσον αφόρα την εκτέλεση του προγράμματος χρησιμοποιούνται οι παρακάτω εντολές:

- make run-lsh → εκτελεί τον αλγόριθμο lsh
- make run-hc → εκτελεί τον αλγόριθμο hupercube
- make run-cluster-classic → εκτελεί το cluster με την χρήση του αλγόριθμου Lloyds
- make run-cluster-lsh → εκτελεί το cluster με την χρήση του lsh
- make run-cluster-hc → εκτελεί το cluster με την χρήση του hypercube
- make clean → διαγράφει τα .o και εκτελέσιμα αρχεία

Ή αλλιώς, αν δεν θέλετε να το τρέξεται με default τιμές
./bin/"exe" -i inputfile -o ouputfile [+ args]

Τα αποτελέσματα των παραπάνω εντολών εμφανίζονται στο directory logs στόν αρχείο logs.txt (logs/logs.txt).

2 Κατάλογος αρχείων κώδικα

Αρχικά το πρόγραμμα έχει οργάνωθει στα εξής διρεστοριες :

- bin → περιέχει τα binary files
- build → περιέχει τα object files (.o) των αρχείων κώδικα
- include → περιέχει όλα τα αρχεία επικεφαλίδες (.h)

- src → περιέχει όλα τα αρχεία κώδικα (.c)

- logs → περιέχει την έξοδο του προγράμματος

- assets → κρατάει τα απαραίτητα δεδομένα για να μεταγλωτιστεί και να εκτελεσθεί το πρόγραμμα

Ειδικότερα για τα αρχεία που υπάρχουν στα directories include src:

- data.h and data.cpp →περιέχει την κλάσση Data , η οποία αποθηκεύει τα δεδομένα και τα queries που θα χρησιμοποιηθούν για την εκτέλεση του προγράμματος καθώς και το πρότυπα των συνάρτησεων που αποθηκεύουν τα δεδομένα αλλα και των συναρτήσεων αναζήτησης και απόστασης μεταξύ των δεδομένων.

- hashTable.h and hashTable.cpp →περιέχει το hashtable που χρησιμοποιείται τόσο στον αλγόριθμο lsh όσο και στον αλγόριθμο hypercube για την αποθήκευση των δεδομένων και παρέχει συναρτήσεις υπολογισμού του a, s, h.

- hyperCube.h and hyperCube.cpp →περιέχει όλες τις πληροφορίες σχετικά με τον υπερκύβο και τις συναρτήσεις του όπως και την κλάσση f , η οποία είναι απαραίτητη για αυτόν. Όσον αφορά την f αυτή χρησιμοποιεί δύο unordered_set ένα για όλες τις τιμές που στέλνει στο μηδέν και ένα για όλες τις τιμές που στέλνει στο 1. Προτιμήθηκαν δύο unordered_set , καθώς αποθηκεύουν μόνο τα κλειδιά και όχι τις τιμές μηδέν και ένα και γλυτώνουμε χώρο , αφού αν φτιάχναμε ένα hashtable για την αποθήκευση όλων θα είχαμε πολλές χιλιάδες byte χαμένα , αποθηκεύοντας συνεχώς το 0 και 1.

- input.h and input.cpp →διαβάζει τα ορίσματα της γραμμής εντολών.

- kmeansplusplus.h and kmeansplusplus.cpp → Περιέχει όλες στις συναρτήσεις και του λογική του cluster. Συναρτήσεις όπως η intializeCentroids και άλλες που αφορούν την ανάθεση των point και την καταμέτρηση ακρίβειας του clustering.

- LSH.h and LSH.cpp →Περιέχει όλες τις συναρτήσεις σχετικά με τον αλγόριθμο lsh. Κανει initialize ολα τα hashtables δημιουργώντας τα s, και στην συνέχεια αποθηκεύει ολα τα δεδόμενα υπολογίζωντας την τιμή g για το κάθε ένα.

- main.cpp → καλεί τις ανάλογες συνάρτησεις ανάλογα με το input.

3 Περιγραφή προγράμματος

Αρχικά το πρόγραμμα δέχεται τις εντολές , τις οποίες χρειάζεται για να τρέξει και σταματάει την εκτέλεση και εκτυπώνει το κατάλληλο μήνυμα αν κατι δεν πάει όπως πρέπει. Ύστερα αποθηκεύει όλα τα δεδομένα σε ένα εστορ . Στη συνέχεια αναλόγως την εντολή του χρήστη κάλει την ανάλογη συνάρτηση είτε για την εκτέλεση του αλγόριθμου lsh ειτε του hypercube ειτε για τα διάφορα cluster.

3.1 LSH

Κατά την αρχικοποίηση της κλάσσης ύστερα απο την δημιουργία στιγμιοτύπου της στην μαιν φυνςτιον καλείται ο constructor , στον οποίο γίνεται η αρχικοποίηση της μεταβλητής M , των L hashtable και εν τέλει καλείται μέσα απο τον constructor και η συνάρτηση hashdata() .Η hashdata() μοιράζει όλα τα δεδομένα σε κάθε ενα απο τα L hashtables με την χρήση της συνάρτησης calculate$_g$() , η οποία δείχνει σε ποιό bucket θα πάει καθε δεδομένο σύμφωνα με τις διαφάνεις. Εν συνεχεία καλείται η συνάρτηση LSH::Run() , η οποία για κάθε query καλεί την exec_query χρονομετρώντας την. Η exec$_q uery$ για κάθε ενα απο τα L hashtables βρίσκει το bucket στο οποίο αντιστοιχεί το query και διαλέγει τους πιθανούς γείτονες , ενω αποκλείει τα διπλότυπα διανύσματα που ενδεχομένως να εμφανιστούν απο hashtable σε hashtable , και τα τοποθετεί σε ενα vector . Στο τέλος γυρνάει ενα vector αφου πρώτα καλέσει την Data::GetClosestNeighbors . Σε αυτήν επιλέγονται οι καταλληλότεροι γείτονες με την βοήθεια ενος queue και επιστρέφει την απόσταση και το index του κάθε γείτονα . Επιστρέφοντας στην LSH::Run() μετα την κλήση της LSH::exec_query , πραγματοποιείται η κλήση της Data::BruteForceNeighbors , η οποία βρίσκει με brute force τρόπο τους N πλησιέστερους γείτονες , ενώ χρονομετρεί και τον χρόνο εκτέλεσης της . Τελικά η LSH::Run() περνάει όλα τα δεδομένα που μάζεψε στη συναρτηση LSH::print() , ώστε να εκτυπώσει τα αποτελέσματα στο output file με την ζητούμενη μορφολογία.

3.2 Hypercube

Κατά την αρχικοποίηση της κλάσσης ύστερα απο την δημιουργία στιγμιοτύπου της στην main function καλείται ο constructor , στον οποίο γίνεται η αρχικοποίηση του hashtable , του πίνακα των f και εν τέλει καλείται μέσα απο τον constructor και η συνάρτηση hashdata() .Η hashdata() μοιράζει όλα τα δεδομένα στο hashtable με την χρήση της συνάρτησης f::calculate$_f$() , η οποία δείχνει σε ποιό bucket θα πάει καθε δεδομένο σύμφωνα με τις διαφάνεις. Εν συνεχεία καλείται η συνάρτηση HyperCube::Run() , η οποία για κάθε query καλεί την exec_query χρονομετρώντας την. Η exec_query βρίσκει το bucket στο οποίο αντιστοιχεί το query και με την χρήση της HyperCube::hammingDist() βρίσκει τον ζητούμενο αριθμό κορυφών (probes) με αύξουσα hamming απόσταση . Αυτο σημαίνει πως πρώτα θα επιλέξει ολα τα buckets με hamming απόσταση 1 και αν δεν έχουμε βρεί ακόμα τόσα όσα μας υπαγορεύει το probes συνεχίζουμε να επιλέγουμε με hamming απόσταση 2 και ούτω καθεξής. Στη συνέχεια διατρέχοντας την λίστα με τα buckets ψάχνουμε M δεδομενα και τα τοποθετούμε σε ενα vector με τους πιθανούς γείτονες . Η διαδικασία αυτή σταματάει όταν βρούμε M γείτονες ή όταν ψάξουμε αριθμό buckets ισο με το probes. Στο τέλος γυρνάει ενα vector αφου πρώτα καλέσει την Data::GetClosestNeighbors . Σε αυτήν επιλέγονται οι καταλληλότεροι γείτονες με την βοήθεια ενος queue και επιστρέφει την απόσταση και το index του κάθε γείτονα . Επιστρέφοντας στην Hypercube::Run() μετα την κλήση της HyperCube::exec_query , πραγματοποιείται η κλήση της Data::BruteForceNeighbors() , η οποία βρίσκει με brute force τρόπο τους N πλησιέστερους γείτονες , ενώ χρονομετρεί και τον χρόνο εκτέλεσης της . Τελικά η HyperCube::Run() περνάει όλα τα δεδομένα που μάζεψε στη συναρτηση Hypercube::print() , ώστε να εκτυπώσει τα αποτελέσματα στο output file με την ζητούμενη μορφολογία.

3.3 Kmeans++

Η κλάσση αυτή περιέχει πολλαπλούς constructor ανάλογα με την λειουργία που επιθυμεί ο χρήστης. Classic, LSH or Hypercube. Αρχικά επιλέγουμε ένα centroid στην τύχη, και τα υπόλοιπα με την τεχνική kmeans++. Όταν κληθεί η συνάρτηση kmeansplusplus::Run(), το πρόγραμμα μπαίνει σε μια επανάληψη όπου σε κάθε βήμα έχει ένα `vector<vector<int>>`, μεγέθους όσο τα σημεία centroids όπου αποθηκεύεται ένας πίνακας με το τα index των δεδομένων που αντιστοιχούν σε κάθε cluster. Η ανάθεση αυτή, στην κλασσική μέθοδο γένται με σύγκριση όλων τον αποστάσεων για κάθε σημείο, με τα centroid.Στην μέθοδο LSH και Hypercube, καλώ την exec_query() συνάρτηση της κάθε κλάσσης με όρισμα τις συντεταγμένες κάθε centroid, και αναθέτω όλα τα σημεία που μου επιστρέφει το αντίστοιχο centroid. Ακόμα χρησιμοποιώ ένα unorder_set όπου κρατάω το index όλων των σημείων που έχω αναθέσει, έτσι στο τέλος, όσα σημεία δεν έχουν ανατεθεί άπλα τα αναθέτω όπως στην κλασσική μέθοδο. Αφού έχει γίνει η ανάθεση, υπολογίζουμε τις μέσεις τιμές ή τις ενδιάμεσες τιμές για κάθε cluster και το μετακινούμε αντίστοιχα. Η επανάληψη τελειώνει όταν έιτε η συνολική μετακίνηση είναι μικρότερη του kmeansplusplus::minChange είτε όταν γίνουν συνολικά kmeansplusplus::maxIterations. Παρατηρήσαμε ότι υπολογίζοντας το την μέση τιμή αντί για την ενδιάμεση είχαμε λίγο καλύτερα αποτελέσματα. Τέλος, ελέγχουμε την ποιότητα του clustering τρέχοντας την συνάρτηση kmeansplusplus::Silouette(), η οποία με βάση τον τύπο στις σημειώσεις μας δίνει ενα αποτέλεσμα για κάθε cluster μεταξύ του -1 και του 1.