Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
Τμήμα Πληροφορικής & Τηλεπικοινωνιών
Ανάπτυξη Λογισμικού για Αλγ. Προβλήματα

Γιώργος Κατσογιάννης-Μεϊμαράκης
sdi1400065@di.uoa.gr
Χειμερινό Εξάμηνο 2018-19

Approximate Nearest Neighboor & Clustering

Περιεχόμενα

1 Μεταγλώττιση & Εκτέλεση

1.1 Μεταγλώττιση

Τα προγράμματα μεταγλωττίζονται με τη χρήση του αρχείου Makefile και συγκεκριμένα τις εντολές:

- `make`
  - Για τη δημιουργία και όλων των εκτελέσιμων
- `make lsh`
  - Για τη δημιουργία μόνο του εκτελέσιμου lsh
- `make cube`
  - Για τη δημιουργία μόνο του εκτελέσιμου cube
- `make cluster`
  - Για τη δημιουργία μόνο του εκτελέσιμου cluster

1.2 Εκτέλεση

1.2.1 LSH

```
./lsh -d <input file> -q <query file> -k <int> -L <int> -o <output file>
```

1.2.2 Παράμετροι LSH

- L είναι το πλήθος των hash tables που θα δημιουργηθούν
- k είναι το πλήθος των συναρτήσεων κατακερματισμού h_i ανά hash table

1.2.3 Hypercube

```
./cube -d <input file> -q <query file> -k <int> -M <int> -p <int> -o <output
  file>
```

1.2.4 Παράμετροι Hypercube

- k είναι η διάσταση του υπερκύβου
- M είναι το μέγιστο πλήθος σημείων που θα ελέγχονται σε κάθε αναζήτηση
- p είναι το μέγιστο πλήθος κορυφών που θα ελέγχονται ανά αναζήτηση

1.2.5 Cluster

```
./cluster -i <input file> -c <configuration file> -o <output file>
```

1.3 Input File

Τα αρχεία εισόδου όλων των προγραμμάτων του project πρέπει να έχουν την ακόλουθη μορφή:

```
x0  0   16  35  5   32  31  14  10  11  78  55  10  45  83  11  6   14  57  ...
x1  14  35  19  20  3   1   13  11  16  119 85  5   0   5   24  26  0   27  ...
x2  0   1   5   3   44  40  20  14  10  100 63  7   44  47  9   6   7   70  ...
x3  12  47  14  25  2   3   4   7   14  122 90  7   0   0   6   14  0   24  ...
...
```

```
x0,0,16,35,5,32,31,14,10,11,78,55,10,45,83,11,6,14,57,...
x1,14,35,19,20,3,1,13,11,16,119,85,5,0,5,24,26,0,27,...
x2,0,1,5,3,44,40,20,14,10,100,63,7,44,47,9,6,7,70,...
x3,12,47,14,25,2,3,4,7,14,122,90,7,0,0,6,14,0,24,...
...
```

Κάθε γραμμή δίνει και ένα σημείο του συνόλου δεδομένων. Για κάθε σημείο πρώτα δίνεταιτο όνομά του και μετά τα στοιχεία του, τα οποία διαχωρίζονται μεταξύ τους με tabs ή κόμματα.

1.4 Configuration File

Το configuration file χρησιμοποιείται για την ρύθμιση των παραμέτρων των προγραμμάτων.

1.4.1 Πιθανές Παράμετροι

Οι παράμετροι που μπορούν να δωθούν μέσω του configuration file είναι οι εξής:

Παράμετρος	Default	Περιγραφή
number_of_clusters: <int>	-	Πλήθος των clusters που θα δημιουργηθούν
number_of_hash_functions: <int>	4	Πλήθος hash functions του LSH
number_of_hash_tables: <int>	5	Πλήθος hash tables του LSH
hypercube_dimension: <int>	5	Διάσταση του Hypercube
hypercube_max_point_checks: <int>	5000	Μέγιστο πλήθος σημείων που ελέγχει ο Hypercube
hypercube_max_vertex_checks: <int>	20	Μέγιστο πλήθος κορυφών που ελέγχει ο Hypercube
max_iterations: <int>	100	Μέγιστο πλήθος επαναλήψεων στο clustering
metric: <string>	euclidean	Μετρική απόστασης (euclidean, cosine)
initialise: <string>	random	Αλγόριθμος αρχικοποίησης στο clustering (random, kmeans)
assign: <string>	lloyds	Αλγόριθμος ανάθεσης στο clustering (lloyds, lsh, hypercube)
update: <string>	kmeans	Αλγόριθμος ανανέωσης συστάδων στο clustering (kmeans, pam)

2 Εργαλεία Ανάπτυξης Project

2.1 Version Control (Git/Github)

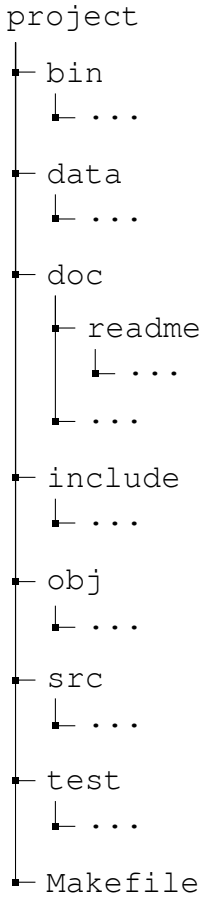
Για την καλύτερη διαχείρηση των εκδόσεων του κώδικα και των αλλαγών χρησιμοποιείται το πρόγραμμα git και η πλατφόρμα Github.

2.2 Unit Testing (CppUnit)

Η εγκατάσταση του CppUnit γίνεται με την εντολή:

```
sudo apt-get install libcppunit-dev
```

3 Οργάνωση Αρχείων & Φακέλων



Ο κώδικας οργανώνεται σε διαφορετικά αρχεία ανάλογα με το σκοπό και τη λειτουργικότητά του. Συγκεκριμένα στα εξής αρχεία:

3.1 Οργάνωση Φακέλων

3.1.1 Φάκελος bin

Περιέχει τα τελικά εκτελέσιμα αρχεία που παράγονται με τη μεταγλώττιση του κώδικα.

3.1.2 Φάκελος data

Περιέχει όλα τα σύνολα δεδομένων που χρησιμοποιούνται.

3.1.3 Φάκελος doc

Περιέχει τα αρχεία που περιγράφουν το project (readme, configuration files).

3.1.4 Φάκελος include

Περιέχει όλα τα αρχεία κεφαλίδας (.hpp) των αρχείων κώδικα που αναπτύχθηκαν για το project.

3.1.5 Φάκελος obj

Περιέχει τα object files που παράγονται κατά τη μεταγλώττιση του κώδικα.

3.1.6 Φάκελος src

Περιέχει όλα τα αρχεία πηγαίου κώδικα (.cpp) που αναπτύχθηκαν για το project.

3.1.7 Φάκελος test

3.2 Οργάνωση Αρχείων Κώδικα

3.2.1 Αρχείο lsh.cpp

Περιέχει την συνάρτηση main που εκτελεί το LSH πάνω σε δοσμένο dataset και εκτυπώνει αποτελέσματα και στατιστικά.

3.2.2 Αρχείο cube.cpp

Περιέχει την συνάρτηση main που εκτελεί τον υπερκύβο πάνω σε δοσμένο dataset και εκτυπώνει αποτελέσματα και στατιστικά.

3.2.3 Αρχείο cluster.cpp

Περιέχει την συνάρτηση main που εκτελεί το cluster creator πάνω σε δοσμένο dataset και εκτυπώνει αποτελέσματα και στατιστικά.

3.2.4 Αρχείο searcher.cpp/.hpp

Περιέχουν τη virtual κλάση searcher και την υλοποίηση των υποκλάσεών της, lsh και hypercube.

3.2.5 Αρχείο clusterCreator.cpp/.hpp

Περιέχουν την υλοποιήση των αλγορίθμων clustering και του αλγορίθμου αξιολόγησης silhouette.

3.2.6 Αρχείο hash_table.cpp/.hpp

Περιέχει την υλοποίηση των πινάκων κατακερματισμού που χρησιμοποιεί ο αλγόριθμος LSH

3.2.7 Αρχείο hasher.cpp/.hpp

Περιέχει την υλοποίηση των μετρικών που χρησιμοποιούνται και από τις δύο μεθόδους (ευκλείδιας απόστασης και συνημιτόνου)

3.2.8 Αρχείο point.cpp/.hpp

Περιέχει την υλοποίηση του ΑΤΔ point το οποίο αναπαριστά τα σημεία των datasets και τα διανύσματα των αλγορίθμων

3.2.9 Αρχείο util.cpp/.hpp

Περιέχει γενικές συναρτήσεις που χρησιμοποιούνται και από τις δύο μεθόδους

4 Approximate Nearest Neighboor

4.1 Latent Semantic Hashing (LSH)

4.2 Υπερκύβος

5 Σύγκριση Μεθόδων LSH & Hypercube

Για την σύγκριση των δύο μεθόδων θα χρησιμοποιήσουμε την **μετρική συνημιτόνου**.

5.1 Ενδεικτικές Εκτελέσεις

Στη συνέχεια βλέπουμε τις επιδόσεις των μεθόδων για διαφορετικές παραμέτρους. Είναι πολύ σημαντικό να σημειώθει ως σημείο αναφοράς ότι ο ντετερμινιστικός αλγόριθμος πλησιέστερου γείτονα τρέχει κατά μέσο όρο σε χρόνο 0.05s. Χρησιμοποιούμε ένα dataset 10000 σημείων ως είσοδο και ένα dataset 100 σημείων για queries.

5.1.1 LSH

k	L	Max Approx.	Avg. Approx	Avg. Time
3	5	1	1	0.2292
4	5	1.0559	1.0011	0.2089
6	5	1.1177	1.0050	0.1469
8	5	1.1070	1.0058	0.0900
10	5	1.5502	1.0367	0.0456
12	5	1.8626	1.0400	0.0416
14	5	2.1293	1.0489	0.0298
15	5	2.1771	1.0786	0.0202
16	5	2.3813	1.0956	0.0274
16	6	2.1134	1.0783	0.0421
16	7	2.0756	1.0808	0.0357
16	8	1.8307	1.0690	0.0235
16	**9**	**1.7638**	**1.0438**	**0.0341**
16	10	1.6221	1.0505	0.0464

5.1.2 Hypercube

k	M	p	Max Approx.	Avg. Approx	Avg. Time
4	500	4	5.0334	1.3621	0.0027
5	500	4	4.9590	1.4644	0.0025
6	500	4	3.3547	1.2262	0.0024
6	1000	4	3.1239	1.1584	0.0050
6	1000	6	1.7797	1.0844	0.0054
6	1000	10	2.4349	1.0989	0.0052
6	2000	10	2.6356	1.0793	0.0069
7	3000	15	2.2929	1.0542	0.0119
7	3000	32	1.7262	1.0447	0.0131
7	**3000**	**64**	**1.6721**	**1.0509**	**0.0138**
5	3000	15	2.2484	1.0857	0.0149
5	3000	32	1.8530	1.0498	0.0150

5.1.3 Παρατηρήσεις & Σχόλια

Από τις παραπάνω εκτελέσεις παρατηρούμε ότι και οι δύο μέθοδοι καταφέρνουν αρκετά καλά αποτελέσματα με σχετικά παρόμοια ακρίβεια (συγκεκριμένα για $k = 16, L = 9$ στο LSH και $k = 7, M = 3000, p = 64$ στον υπερκύβο). Παρατηρούμε όμως ότι η μέθοδος του υπερκύβου είναι περίπου 2 φορές πιο γρήγορη από το LSH και περίπου 3 φορές πιο γρήγορη από τον ντετερμινιστικό αλγόριθμο.

Επίσης, για το LSH παρατηρούμε ότι δεν έχει νόημα η παράμετρος k να παίρνει τιμές κάτω από 14 για τη συγκεκριμένη μετρική, διότι ο χρόνος που πετυχαίνει είναι χειρότερος από τον ντετερμινιστικό αλγόριθμο. Αυτό συμβαίνει διότι η συνάρτηση κατακερματισμού έχει πολλές συγκρούσεις για μικρά k.

Ακόμη, στον υπερκύβο παρατηρούμε ότι είναι σημαντικό όταν αυξάνεται το k να αυξάνεται και το p, το οποίο είναι λογικό αφού όταν έχουμε πολλές κορυφές θα πρέπει να επιτρέπουμε στον αλγόριθμο να επισκέπτεται περισσότερες.

5.2 Κατανάλωση Χώρου

5.2.1 LSH

Για την υλοποίηση του LSH χρησιμοποιούνται L hash tables, καθένα απο τα οποία χρησιμοποιεί τη δική του συνάρτηση κατακερματισμού και κάθε μια απ' αυτές έχει k "υπο-συναρτήσεις" με ξεχωριστά διανύσματα ανάλογα τη μετρική.

Για κάθε hash table δημιουργούνται k buckets, τα οποία αναπαρίστανται με C++ vectors και σε αυτά αποθηκεύονται οι δείκτες των σημείων που είναι αποθηκευμένα σε μια εξωτερική δομή για την αποφυγή της πολλαπλής (για την ακρίβεια k-πλής) αποθήκευσης των σημείων.

5.2.2 Υπερκύβος

Στην περίπτωση του υπερκύβου, έχουμε 2^k κορυφές οι οποίες έχουν ίδια μορφή με τα buckets των hash tables (C++ vectors με pointers σε σημεία), μόνο μια συνάρτηση κατακερματισμού με 2^k υποσυναρτήσεις και k συναρτήσεις $f : x \rightarrow [f_1(h_1(x)), \ldots, f_{2^k}(h_{2^k}(x))) \in \{0,1\}^{2^k}]$. Για την υλοποίηση των συναρτήσεων f χρησιμοποιείται η δομή unordered_map της C++ όπου κάθε τιμή που δεν περιέχεται ήδη στο map προβάλεται τυχαία στο $\{0,1\}$.

6 Σύγκριση Μεθόδων Clustering

6.1 Σύγκριση Αλγορίθμων Αρχικοποίησης

Συγκρίνουμε τους αλγορίθμους αρχικοποίησης τυχαίας επιλογής την παραλαγή του k-means. Οι συγκρίσεις γίνονται στο μικρό dataset χρησιμοποιώντας την μετρική ευκλείδιας απόστασης.

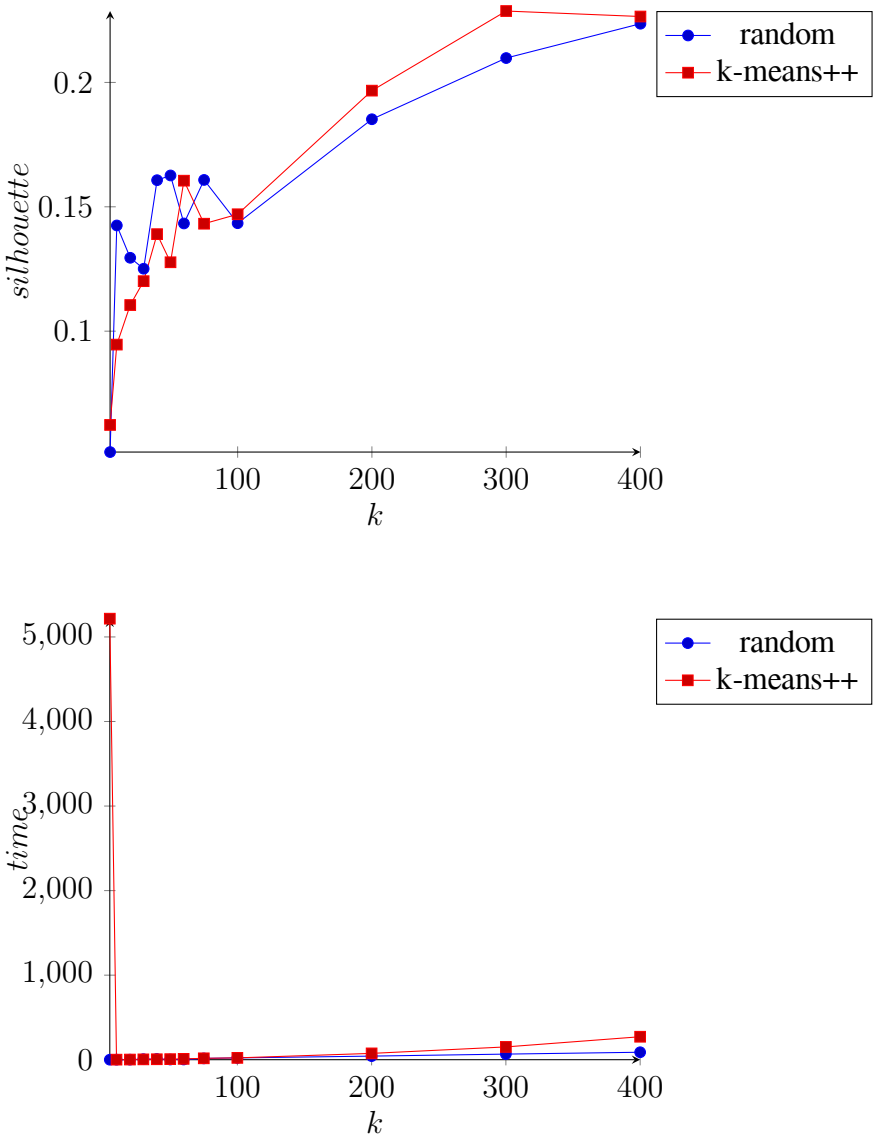
Παρατηρούμε πως δεν υπάρχει σαφής διαφορά στο αποτέλεσμα που παίρνουμε και συνήθως οι ποιότητα των clusters είναι σχεδόν ίδια. Επίσης παρατηρούμε ότι όσο αυξάνεται το k, ο χρόνος που χρειάζεται η αρχικοποίηση του k-means++ γίνεται όλο και μεγαλύτερη της τυχαίας.

k	Time	Avg. Silhouette
5	0.1842	0.0514
10	0.6250	0.1425
20	0.8671	0.1295
30	6.2896	0.1251
40	8.3453	0.1607
50	2.3731	0.1626
60	2.6831	0.1433
75	15.5488	0.1608
100	21.4363	0.1434
200	42.6529	0.1852
300	66.2964	0.2098
400	88.3738	0.2236

Random & Lloyd's & k-Means

k	Time	Avg. Silhouette
5	05216	0.0623
10	0.7706	0.0946
20	1.9548	0.1105
30	4.2070	0.1201
40	5.4749	0.1390
50	6.6362	0.1277
60	10.2781	0.1605
75	16.5979	0.1432
100	21.8534	0.1470
200	74.8394	0.1967
300	151.3390	0.2287
400	271.7470	0.2265

k-Means++ & Lloyd's & k-Means



6.2 Σύγκριση Αλγορίθμων Ανάθεσης

Συγκρίνουμε τους αλγορίθμους ανάθεσης με range search χρησιμοποιώντα LSH και τον υπερκύβο. Οι συγκρίσεις γίνονται στο μικρό dataset χρησιμοποιώντας την μετρική ευκλείδιας απόστασης. Για το LSH χρησιμοποιούμε $k = 16$ hash function και $L = 9$ hash tables. Για τον υπερκύβο χρησιμοποιούμε $k = 7$ για διάσταση του υπερκύβου, ελέγχουμε το πολύ $M = 5000$ σημεία και το πολύ $p = 64$ ακμές του υπερκύβου.

6.3 Σύγκριση Αλγορίθμων Ανανέωσης

Συγκρίνουμε τους αλγορίθμους ανανέωσης k-means και partition arround mediod (PAM). Οι συγκρίσεις γίνονται στο μικρό dataset χρησιμοποιώντας την μετρική ευκλείδιας απόστασης.

Παρατητούμε πως ο αλγόριθμος PAM καταφέρνει αρκετά καλύτερα αποτελέσματα απ' ό,τι ο k-means σχεδον για όλα τα k. Επίσης, αν και για μικρά k ο PAM είναι αργότερος του k-means, όσο μεγαλώνει το k, ο PAM γίνεται πιο γρήγορος, πετυχαίνοντας τεράστιες διαφορές για μεγάλα k.

k	Time	Avg. Silhouette
5	0.1842	0.0514
10	0.6250	0.1425
20	0.8671	0.1295
30	6.2896	0.1251
40	8.3453	0.1607
50	2.3731	0.1626
60	2.6831	0.1433
75	15.5488	0.1608
100	21.4363	0.1434
200	42.6529	0.1852
300	66.2964	0.2098
400	88.3738	0.2236

Random & Lloyd's & k-Means

k	Time	Avg. Silhouette
5	31.282	0.0767
10	28.4778	0.1671
20	30.2874	0.2793
30	24.0356	0.2467
40	34.1091	0.2591
50	22.856	0.2701
60	27.1966	0.2676
75	30.9813	0.2808
100	26.6153	0.2726
200	9.9055	0.2721
300	9.4906	0.2715
400	6.5673	0.2838

Random & Lloyd's & PAM