Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα

2η Προγραμματιστική Εργασία Κατακερματισμός και αναζήτηση για χρονοσειρές στη C++

Μηνάς Διολέτης (Α.Μ. 1115201400272)
Απόστολος Θεοδώρου (Α.Μ.: 1115201500046)

Χειμερινό Εξάμηνο 2021-2022
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Contents

1 Περιγραφή της εργασίας

Σκοπός της παρούσας εργασίας είναι η υλοποίηση αλγορίθμων εύρεσης πλησιέστερου γείτονα χρονοσειράς και συσταδοποίησης συνόλου χρονοσειρών σε γλώσσα C++.

Το πρόγραμμα εύρεσης κοντινότερου γείτονα χρονοσειράς(search) λαμβάνει ως είσοδο αρχείο που περιέχει τις χρονοσειρές των τιμών κάποιων μετοχών του χρηματιστηρίου Nasdaq (input_file) και ένα δεύτερο αρχείο που επίσης περιέχει χρονοσειρές μετοχών (query_file). Οι χρονοσειρές του πρώτου αρχείου αποτελούν το σύνολο δεδομένων (dataset), ενώ αυτές του δεύτερου το σύνολο αναζήτησης. Στόχος είναι να παραχθεί ως έξοδος ένα αρχείο, στο οποίο, ανάλογα με τις δοθείσες παραμέτρους, να επιστρέφεται η κοντινότερη χρονοσειρά του dataset για κάθε μία από τις χρονοσειρές του συνόλου αναζήτησης.

Το πρόγραμμα της συσταδοποίησης (cluster) λαμβάνει, όπως και το προηγούμενο πρόγραμμα, ένα αρχείο που περιέχει το dataset. Επιπλέον δέχεται και ένα αρχείο (configuration_file) στο οποίο περιέχονται πληροφορίες για την παραμετροποίηση της συσταδοποίησης. Η έξοδός του είναι ένα αρχείο που περιλαμβάνει πληροφορίες για τις συστάδες χρονοσειρών που δημιουργήθηκαν.

Οι αλγόριθμοι που υλοποιούν τα προγράμματα της εύρεσης κοντινότερων γειτόνων είναι ο lsh (Locality Sensitive Hashing) ο αλγόριθμος τυχαίας προβολής στον υπερκύβο και ο αλγόριθμος Frechet με δύο παραλαγγές (discrete, continuous) . Το πρόγραμμα της συσταδοποίησης δύναται να εκτελεστεί με τις εξής εναλλακτικές μεθόδους ανάθεσης καμπυλών σε συστάδες:

- τον ακριβή αλγόριθμο του Lloyds
- την αντίστροφη ανάθεση (reverse) μέσω Range Search
  - με lsh.
  - με τυχαία προβολή
  - με Frechet

Επίσης δύναται να εκτελεστεί για διαφορετικές μετρικές απόστασης (ευκλε-

ίδεια απόσταση και διακριτή ή συνεχής απόσταση) και τεχνικές επανάθεσης σε νέα συστάδα. Δείτε αναλυτικά την εκφώνηση της εργασίας

2 Αρχεία

Τα προγράμματα αναπτύχθηκαν σύμφωνα με τις αρχές του modular programming και ο κώδικας, αν και εκτενής, είναι διαρθρωμένος σε ξεχωριστά αρχεία, ανάλογα με τη λειτουργικότητα που επιτελεί η κάθε συνάρτηση. Ακολουθεί λίστα των αρχείων και περιγραφή τους.

Η main και οι συναρτήσεις της:

- main.cpp: περιέχει τη συνάρτηση main που υλοποιεί τη λειτουργικότητα κάθε προγράμματος καλώντας τις κατάλληλες συναρτήσεις, ανάλογα με τις παραμέτρους της γραμμής εντολών ή/και του configuration file.
- cmd_line_args.cpp: υλοποιεί την ανάθεση των παραμέτρων της γραμμής εντολών στις αντίστοιχες μεταβλητές.
- conf_file.cpp: υλοποιεί το διάβασμα του configuration file του cluster και την αρχικοποίηση των αντίστοιχων μεταβλητών.
- file_functions.cpp: χρησιμοποιείται για το άνοιγμα αρχείων (input, query, output και configuration) , το διάβασμα και το γράψιμο και από και προς αυτά, το κλείσιμο του μετά το πέρας του προγράμματος.
- user_input.cpp: διαχειρίζεται την ανάδραση του τον χρήστη στην περίπτωση του ή/και hypercube όταν ζητείται η επανάληψη εκτέλεσης του προγράμματος για διαφορετικό σύνολο (αρχείο) αναζήτησης.

Συναρτήσεις γενικού σκοπού:

- hamming_distance.cpp: υλοποιεί την εύρεση της Χάμινγκ απόστασης μεταξύ δυαδικών συμβολοσειρών, χρησιμοποιείται στην μέθοδο του υπερκύβου.

- mod.cpp: υλοποιεί την εύρεση της ομώνυμης πράξης (επιστρέφοντας μόνο θετικά υπόλοιπα).
- vector_ops.cpp: περιέχει όλες τις συναρτήσεις που αφορούν πίνακες (vectors)και χρησιμοποιούνται από πολλές συναρτήσεις τόσο της εύρεσης γειτόνων, όσο και τις συσταδοποίησης. Λεπτομέρειες για τον σκοπό της κάθε συνάρτησης θα βρείτε σε σχόλιο πάνω από τον ορισμό της.

Για το πρόγραμμα search για τους αλγορίθμους lsh και hypercube:

- lsh.cpp: περιέχει τις υλοποιήσεις των αλγορίθμων lsh, range_search με lsh και brute force knn.
- cube.cpp: περιέχει τις υλοποιήσεις της τυχαίας προβολής στον υπερκύβο και range search με hypercube.
- hashTable.cpp:υλοποιεί τους πίνακες κατακερματισμού των μεθόδων hypercube και lsh.
- hash_functions.cpp: υλοποιεί τις συναρτήσεις κατακερματισμού h, g που χρησιμοποιούνται στην καταχώρηση σημείων στους πίνακες και στην εύρεση των id στις περιπτώσεις που απαιτείται.
- knn_table_functions.cpp: περιέχει χρήσιμες βοηθητικές συναρτήσεις (ταξινόμησης, αναζήτησης κ.α.) για τον αλγόριθμο της εύρεσης κοντινότερων γειτόνων.

Για το πρόγραμμα search για τον αλγόριθμο Frechet:

- frechet_functions.cpp:υλοποιεί τους αλγορίθμους εύρεσης απόστασης και γειτονικών χρονοσειρών, σύμφωνα με τον αλγόριθμο Frechet

Για το πρόγραμμα cluster:

- cluster.cpp: περιέχει τις υλοποιήσεις των αλγορίθμων Lloyds, reverse range search with LSH_Vector, reverse range search with LSH_Frechet, reverse range search with hypercube

- lloyds_auxiliary.cpp: περιέχει βοηθητικές συναρτήσης για τον αλγόριθμο του Lloyds.
- silhouette.cpp: υλοποιεί τον υπολογισμό και την εμφάνιση της μετρικής της σιλουέτας.
- mean_curve.cpp: υλοποιεί την update μέθοδο Mean_Frechet με την δενδρική δομή

3 Μεταγλώττιση - Εκτέλεση

Τα προγράμματα "cluster", "search" μεταγλωττίζονται και παράγονται ταυτόχρονα μέσω της εντολής make από το command line, όντας στον κατάλογο ρίζα.

Για το unit testing μεταβείτε στον κατάλογο unit_testing και μεταγλωττίστε με την εντολή make. Έχουμε χρησιμοποιήσει το framework cppunit ,συνεπώς θα πρέπει να έχετε στο σύστημά σας την αντίστοιχη βιβλιοθήκη

Για την εκτέλεση του κάθε προγράμματος τρέξτε τις εντολές που δίνονται στην εκφώνηση. Σε περίπτωση που δεν δώσετε κάποιο από τα μη υποχρεωτικά ορίσματα, αυτό θα αρχικοποιηθεί από τις default τιμές.

Υποχρεωτικά ορίσματα για το search: -algorithm algorithm

Υποχρεωτικά ορίσματα για ./cluster: -c configuration_file, -assignment assignment_method , -update update_method (δώστε την update method με τις λέξεις να χωρίζονται με κάτω παύλα, δηλαδή "Mean_Vector" ή "Mean_Frechet"). Τα προαιρετικά ορίσματα -silhouette, -complete πρέπει να δίνονται πάντα τελευταία.

Για την ανάπτυξη του λογισμικού χρησιμοποιήθηκε το εργαλείο git και η πλατφόρμα github.

Github repository

4 Πειραματική Μελέτη

LSH - SEARCHING

k	L	MAF	$appr.time(\mu s)$	$acc.time(\mu s)$
4	4	1.642	1312.57	2813.29
4	5	1.597	2023.64	3052.64
4	6	1.232	2259.14	2852.71
4	10	1.000	2753.14	3265.43
4	11	1.824	3266.93	2941.64

FRECHET_DISCRETE - SEARCHING

L	δ	MAF	$appr.time$(s)	$acc.time$(s)
5	0.7	3.056	3.010	9.049
7	0.7	2.083	4.59	8.94
10	0.7	2.249	6.72	8.93
5	3	1.910	2.98	8.97
13	6	1.412	6.87	9.19

LLOYD'S CLUSTERING (UPDATE: MEAN_VECTOR)

#cluster	*clust.time*	*silhouette*
3	0	0.3894
4	0	0.3632
5	0	0.3381
7	0	0.2553
10	0	0.4891

LSH - CLUSTERING (UPDATE: MEAN_VECTOR)

k	*L*	*#cluster*	*clust.time*	*silhouette*
4	10	3	0	0.2362
4	10	4	0	0.0401
4	10	5	0	0.2189
4	10	7	0	0.3088
4	10	10	0	0.0012

HYPERCUBE - CLUSTERING (UPDATE: MEAN_VECTOR)

probes	*d'*	*#cluster*	*clust.time*	*silhouette*
2	6	3	0	0.2645
3	6	3	0	0.2597
3	6	5	0	0.1849
4	7	7	0	0.3775
5	7	10	0	0.1434

5 Γενικά Συμπεράσματα

Κάποια γενικά συμπεράσματα στα οποία καταλήξαμε από την πειραματική εκτέλεση των αλγορίθμων για διαφορετικές τιμές στις παραμέτρους τους είναι τα εξής:

Για την αναζήτηση πλησιέστερης χρονοσειράς:

- Για το δοθέν σύνολο δεδομένων παρατηρούμε ότι ένα καλό πλήθος πινάκων κατακερματισμού είναι οι 10. Για λιγότερους πίνακες κατακερματισμού κερδίζουμε μεν χρονικά, αλλά πετυχαίνουμε μικρότερη ακτίβεια. Για περισσότερους από 10 πίνακες αυξάνεται τόσο ο χρόνος υπολογισμού που ο εξαντλητικός αλγόριθμος βρίσκει τους ακριβείς γείτονες νωρίτερα από τον lsh.

- Για τον αλγόριθμο LSH Continuous Frechet ο χρόνος για την εύρεση του προσεγγιστικού γείτονα για μία μόνο καμπύλη κυμάνθηκε στα 82 seconds ενώ ο αντίστοιχος χρός εύρεσης της πραγματικής πλησιέστερης καμπύλης ήταν 330-350 seconds

Για την συσταδοποίηση:

- Ο αλγόριθμος του Lloyd πετυχαίνει τις καλύτερες σιλουέτες σε σχέση με όλους τους άλλους αλγορίθμους που δοκιμάσαμε και, αν και εξαντλητικός, τρέχει στους ίδιους χρόνους με τους προσεγγιστικούς (για μέθοδο update Mean_Vector. Συνεπώς για στη συσταδοποίηση αναδεικνύεται ως ο καταλληλότερος αλγόριθμος.

- Αν και ο αριθμός των συστάδων έχει προεξέχοντα ρόλο στη συσταδοποίηση, παρατηρούμε ότι την ποιότητά της καθορίζουν και άλλες παράμετροι, όπως ο αριθμός των πινάκων κατακερματισμού ή το probes.

- Η συσταδοποίηση με την τεχνική εύρεσης απόστασης discrete Frechet διήρκησε σε μία εκτέλεση 233 seconds και απέδωσε σιλουέτα 0.7158