Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Τελική Αναφορά

Μέλη της Ομάδας: Παναγιώτης Κεχρινιώτης, Αθηνά Χειλάκου
sdi: 202000077,202000213

Μάθημα: Ανάπτυξη Λογισμικού για Πληροφοριακά Συστήματα
Εξάμνο: Χειμερινό Εξάμηνο *2023*

Contents

1. Σύνοψη της Εργασίας

Στο πλάσιο του μαθήματος Ανάπτυξη Λογισμικου για Πληροφοριακά Συστήματα κληθήκαμε να υλοποιήσουμε βελτιστοποιήσεις του αλγορίθμου NNDescent της εύρεσης των N κοντινότερων γειτόνων των στοιχείων ενός γράφου. Η εργασία μας διαρθρώνεται σε τρία παραδοτέα στα οποία επιχειρούμε να προσθέσουμε διαφορετικές βελτιώσεις στον αλγόριθμο.
Η δουλειά μας βασίστηκε κυρίως στην δημοσίευση[1] των Wei Dong, Moses Charikar και Kai Li. Επιλέξαμε να χρησιμοποιήσουμε τη γλώσσα C.

2. Προδιαγραφές

- χειρισμός δεδομένων 100 ή 3 διαστάσεων
- χειρισμός δεδομένων αυθαίρετου αριθμού αντικειμένων
- δυνατότητα εύρεσης των k εγγύτερων γειτόνων για ένα ή για όλα τα μέλη του συνόλου
- δυνατότητα χρήσης εναλλακτικής μετρικής ομοιότητας (ευκλείδεια, Manhattan απόσταση)

3. Πρώτο Παραδοτέο

3.1. Περίληψη

Σε αυτό το παραδότεο υλοποιήσαμε τον βασικό αλγόριθμο NN-descent, ο οποίος βασίζεται στην εξής αρχή: ο γείτονας ενός γείτονα είναι αρκετά πιθανό να είναι επίσης γείτονας. Με άλλα λόγια, αν έχουμε μία προσέγγιση των K εγγύτερων γειτόνων σε ένα σημείο, τότε μπορούμε να βελτιώσουμε την προσέγγισή μας, εξερευνώντας για κάθε σημείο τους γείτονες των γειτόνων, όπως ορίζονται από την τρέχουσα προσέγγιση.

3.2. Περιγραφή

Το παραδοτέο αυτό περιλαμβάνει τρία σκέλη: τον κύριο αλγόριθμο στο αρχείο main ο οποίος βρίσκει για κάθε σημείο των δεδομένων μια προσέγγιση για τους k κοντινότερους γειτονές του, την brute force μέθοδο για την εύρεση της παραμέτρου recall (για την αξιολόγηση του αλγορίθμου) και τη μέθοδο search, για την εύρεση των k κοντινότερων γειτόνων ενός σημείου που δεν ανήκει στο dataset.
Η αρχικοποίηση του KNN γράφου γίνεται τυχαία και οι γείτονες αποθκεύονται σε ενα πίνακα data_size*data_size
Οι k εγγύτεροι γείτονες αποθηκεύονται σε heaps για την αποδοτική τροποποίηση τους κατά την εκτέλεση του αλγορίθμου.

3.3. Παρατηρήσεις/Επισημάνσεις

Ακόμη και χωρίς καμία βελτιστοποίηση ο αλγόριθμος NNDescent έχει υψηλό recall (πλησιάζει το 98% όπως αναμενοταν από τη μελέτη). Η εκτέλεση του όμως διαρκεί πολύ περισσότερο από το brute force (επίσης αναμενόμενο εφόσον δεν χρησιμοποιείται κάποια βελτιώση για την χωρική/χρονική απόδοση του αλγορίθμου).

Παναγιώτης Κεχρινιώτης, Αθηνά Χειλάκου
sdi: 202000077,202000213

4. Δεύτερο Παραδοτέο

4.1. Περίληψη

Σε αυτό το παραδότεο υλοποιήσαμε τις βελτιστοποιήσεις Local Join, Σταδιακή αναζήτηση (Incremental Search), Δειγματοληψία και Πρόωρος Τερματισμός.

4.2. Περιγραφή

4.2.1. Local Join. Αφού όλοι οι υπολογισμοί πραγματοποιούνται κατά μήκος των δύο ακμών του (κόκκινου) γείτονα, μπορούμε να αντιστρέψουμε την οπτική μας και να εστιάσουμε σε αυτόν. Από την οπτική του κόκκινου κόμβου, ο στόχος είναι να εξετάσουμε αν ο γαλάζιος και ο πράσινος κόμβος θα πρέπει να προστεθούν καθένας στη λίστα των k εγγύτερων γειτόνων του άλλου. Με τον τρόπο αυτό αποφεύγουμε τη διπλή επεξεργασία καθώς εξετάζουμε τους γείτονες γειτόνων του γαλάζιου, αλλά και τους γείτονες γειτόνων του πράσινου.
Για την υλοποιήση του local join χρειάστηκε να τροποποιήσουμε σημαντικά τη λογική του

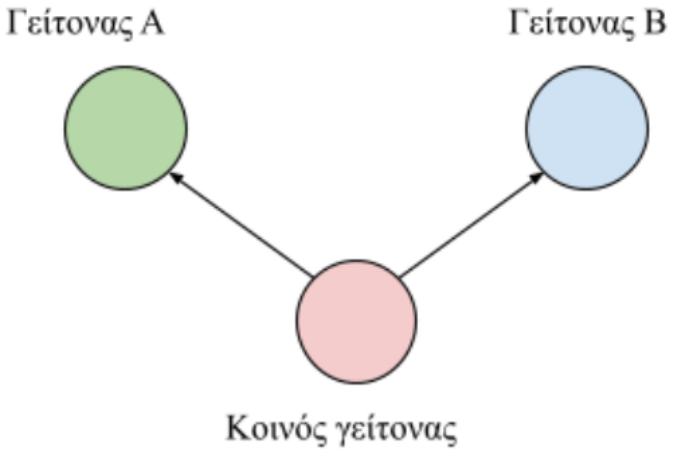

Figure 1: Local Join

κυρίως αλγορίθμου. Χρησιμοποιήσαμε μια δομή map για την αποθήκευση των αντίστροφων γειτόνων κάθε κόμβου, πράγμα μη αποδοτικό λόγω της αυξημένης επιβάρυνσης στη μνήμη που προκαλούσε η αποθήκευση αυτης της δομής για μεγάλα dataset. Ως αποτέλεσμα το πρόγραμμα γινόταν kill από το σύστημα για dataset με 10000 και περισσότερα στοιχεία. Το πρόβλημα αυτο διορθώθηκε στο επόμενο παραδοτέο.

4.2.2. Incremental Search. Καθώς εκτελείται ο αλγόριθμος, όλο και λιγότεροι κόμβοι θα προσθαφαιρούνται στον KNN γράφο σε κάθε επανάληψη. Είναι άσκοπο να πραγματοποιείται ένα πλήρες τοπικό join σε κάθε επανάληψη, αφού αρκετά ζευγάρια έχουν ήδη συγκριθεί σε προηγούμενες επαναλήψεις. Μπορεί να χρησιμοποιηθεί η εξής στρατηγική σταδιακής αναζήτησης για να αποφευχθούν περιττοί υπολογισμοί:

- Προσθήκη μιας **boolean** σημαίας σε κάθε κόμβο στις KNN λίστες. Η σημαία έχει αρχικά την τιμή **true** όταν ο κόμβος εισέρχεται στη λίστα.
- Στο τοπικό **join**, δύο κόμβοι συγκρίνονται μόνο αν τουλάχιστον ένας από τους 2 είναι νέος. Μετά τη συμμετοχή του κόμβου σε ένα τοπικό **join**, η σημαία παίρνει την τιμή **false**.

Για την υλοποίηση του **incremental search** προσθέσαμε **bolean flags** στις ήδη υπάρχουσες δομές **heap** και **map**.

4.2.3. Δειγματοληψία. Πριν από το τοπικό **join**, δειγματοληπτούμε pK, p∈[0,1] κόμβους από τους συνολικούς κόμβους που έχουν τη σημαία **true** και επομένως αναμενόταν να χρησιμοποιηθούν στις συγκρίσεις. Οι αντίστροφες KNN λίστες κατασκευάζονται ξεχωριστά. Οι λίστες αυτές δειγματοληπτούνται με τον ίδιο τρόπο, επομένως η κάθε μία θα έχει το πολύ pK κόμβους. Μόνο οι συγκεκριμένοι κόμβοι που επελέγησαν θα σημανθούν ως **false** στις επόμενες επαναλήψεις.
Υλοποιήσαμε την δειγματοληψία προσθέτοντας ενα όρισμα γραμμής εντολών που καθορίζει τον παράγοντα p.

4.2.4. Πρόωρος τερματισμός. Το κριτήριο φυσικού τερματισμού είναι όταν ο KNN γράφος δεν μπορεί πλέον να βελτιωθεί. Στην πράξη, αριθμός των ενημερώσεων του KNNG μειώνεται δραστικά μετά από κάθε επανάληψη. Στις τελευταίες επαναλήψεις, είναι πιθανόν να μην πραγματοποιείται υπολογιστική εργασία, αλλά κυρίως διαχείριση των εσωτερικών δομών. Μπορούν να χρησιμοποιηθούν τα ακόλουθα κριτήρια πρόωρου τερματισμού για να τερματίσει ο αλγόριθμός, όταν οι επιπλέον επαναλήψεις δεν αναμένεται να επιφέρουν σημαντική βελτίωση στην ακρίβεια: μετράται ο αριθμός των ενημερώσεων στις KNN λίστες σε κάθε επανάληψη και τερματίζεται ο αλγόριθμος όταν πέσει κάτω από δKN, όπου δ μία παράμετρος ακρίβειας (περίπου όσο το κλάσμα των αληθινών KNN που επιτρέπεται να απωλεσθεί λόγω πρόωρου τερματισμού).
Υλοποιήσαμε την πρόωρο τερματισμό προσθέτοντας ενα όρισμα γραμμής εντολών που καθορίζει τον παράγοντα δ και τροποποιώντας κατάλληλα την κύρια επανάληψη του αλγορίθμου.

4.3. Παρατηρήσεις/Επισημάνσεις

Μετά την υλοποίηση όλων των βελτιστοποιήσεων αναμέναμε σημαντική μείωση του χρόνου και του χώρου κατά την εκτέλεση (με μικρό κόστος στην απόδοση του αλγορίθμου στις περιπτώσεις εφαρμογής πρόωρου τερματισμού).Παρ'όλα αυτά η χρήση **maps** για την αποθήκευση των αντιστρόφων γειτόνων προκάλεσε σημαντική επιβάρυνση τόσο στον χώρο που καταλάμβανε το πρόγραμμα όσο και στον χρόνο (συνεχή **hash/rehash** της δομής και δέσμευση/αποδέσμευση μνήμης στο σωρό). Έτσι αντικαταστήσαμε τα **maps** με τα πολύ αποδοτικότερα **avl trees**, περιορίζοντας τη δέσμευση και αποδέσμευση μνήμης στο σωρό κατα την εκτέλεση του προγράμματος σημαντικά. Η αλλαγή αυτή βελτίωσε σημαντικά των αλγόριθμο, ειδικά για μεγαλύτερα **datasets**, χωρίς όμως να επιτύχουμε χρόνο καλύτερο του **brute force**.

5. Τρίτο Παραδοτέο

5.1. Περίληψη

Στο τρίτο παραδοτέο ασχοληθήκαμε με ακόμα τρεις βελτιστοποιήσεις: Βελτιστοποίηση υπολογισμού απόστασης, Αρχικοποίηση KNN γράφου με δένδρα τυχαίων προβολών, Παράλληλη Επεξεργασία.

5.2. Περιγραφή

5.2.1. Βελτιστοποίηση υπολογισμού απόστασης. Για τους σκοπους του αλγορίθμου δεν μας ενδιαφέρει ιδιαίτερα η απόλυτη τιμή της απόστασης, αλλά η σύγκρισήτης με άλλες αποστάσεις. Επομένως, η ακριβή πράξη του υπολογισμού της τετραγωνικής ρίζας δεν είναι απαραίτητη στην ευκλειδια απόσταση και μπορει να παραληφθεί. Έκτος αυτού μπορούμε να μετασχηματισουμε τον τύπο που απομένει ως εξής:

Από την διωνυμική εξίσωση

$$ds_{\mathbf{x},\mathbf{y}} = (x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_N - y_N)^2$$

$$ds = (\mathbf{x} - \mathbf{y})^2$$

$$ds = \mathbf{x}^2 + \mathbf{y}^2 - 2\mathbf{xy}$$

Figure 2: optimal distance

Έτσι οι τετραγωνικές νόρμες των στοιχείων μπορούν να υπολογιστούν στην αρχή και να αποθηκευτούν για χρήση αργότερα.
Για την υλοποίηση της βελτιστοποίησης αυτής προσθέσαμε μια συνάρτηση απόστασης και μια ακόμα που επιστρέφει τον πίνακα με τις τετραγωνικές νόρμες κατά την αρχικοποίηση του αλγορίθμου. Με τη χρήση της βελτιστοποίησης παρατηρήσαμε μια ελάχιστη βελτίωση στον χρόνο εκτέλεσης.

5.2.2. Αρχικοποίηση ***KNN*** γράφου με δένδρα τυχαίων προβολών. Τα δένδρα τυχαίων προβολών (random projection trees) είναι μία αποδοτική τεχνική για την προσεγγιστική αναζήτηση εγγύτερων γειτόνων σε χωρικά δεδομένα (δεδομένα που αναπαριστώνται με διανύσματα). Η ιδέα είναι απλή: ξεκινάμε με την επιλογή ενός τυχαίου υπερεπιπέδου (hyperplane) που χωρίζει τα σημεία σε δύο σύνολα. Στη συνέχεια, αναδρομικά χωρίζεται το κάθε σύνολο με τον ίδιο τρόπο σε μικρότερους υπερκύβους, μέσω επιλογής διαδοχικών τυχαίων υπερεπιπέδων. Η δημιουργία του δένδρου ολοκληρώνεται όταν το μέγεθος του παραγόμενου συνόλου είναι μικρότερος ή ίσος από ένα αριθμό D. Με τον τρόπο αυτό παράγεται ένα δένδρο τυχαίων προβολών. Στο τέλος μπορούμε να υπολογίσουμε το συνολο των αποστάσεων μεταξύ όλων των στοιχείων ενός φύλλου του δένδρου και να τα χρησιμοποιήσουμε για την αρχικοποίηση των ακμών του KNN γράφου.
Για την υλοποίηση της βελτιστοποίησης αυτής δημιουργήσαμε ένα module με όλες τις απαραίτητες συναρτήσεις για την δημιουργία ενός random projection tree. Το module χρησιμοποιείται ως εξής: αρχικά παράγουμε ενα δένδρο τυχαιων προβολών. Όπως συζητήθηκε στο μάθημα ο αριθμός των στοιχείων στα φύλλα είναι αυστηρά μικρότερος από το πλήθος γειτόνων k έτσι ώστε κατά την αρχικοποίηση του γράφου να υπάρχει ένα πλήθος γειτόνων που είναι τυχαίοι για κάθε στοιχείο του dataset, κάτι το οποίο θα εξασφαλίσει ότι ο αλγόριθμος δεν θα τερματίσει πρόωρα λόγω παγίδευσης σε τοπικά βέλτιστα.

Στη συνέχεια αργικοποιούμε τον KNN γράφο βάσει του δένδορου τυχαίων προβολών και εκτελούμε κανονκά τον αλγόριθμο.
Η βελτίωση της χρονικής απόδοσης που παρατηρήσαμε ήταν ελάχιστη το οποίο εικάζουμε ότι οφείλεται στην δημιουργία μόνο ενός random projection tree. Δοθέντος περισσότερου χρόνου, θα είχαμε αξιοποιήσει τεχνικές παραλληλοποίησης για την δημιουργία και τον συνδυασμό πολλαπλών τέτοιων δένδρων.

5.2.3. Παράλληλη Επεξεργασία. Αξιοποιήσαμε τεχνικές παραλληλοποίησης για να επιταχύνουμε την δημιουργία των δένδρων τυχαίων προβολών με τη χρήση της βιβλιοθήκης omp.

5.3. Παρατηρήσεις/Επισημάνσεις

Συνολικά, η απόδοση του αλγορίθμου βελτιώθηκε αρκετά και η βελτίωση αυτή γίνεται αισθητή σε κυρίως μεγαλύτερα datasets.

6. Συμπεράσματα και Πειράματα

6.1. Αποτελέσματα Πειραμάτων

Όλα τα πειράματα εκτελέστηκαν σε λάπτοπ με επεξεργαστή Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz και RAM 8,00 GB.

k	filename	metric	data_type	delta	sampling rate	rpt_init	CPU_time	Recall
100	1000-1.bin	eucl	contest	-	-	-	27.31	1
100	1000-1.bin	manh	contest	-	-	-	23.95	1
100	5000-1.bin	manh	contest	-	-	-	147.73	1

Table 1: Πρώτο Παραδοτέο

k	filename	metric	data_type	delta	sampling rate	rpt_init	CPU_time	Recall
100	1000-1.bin	eucl	contest	0.001	0.4	-	20.58	1
100	1000-1.bin	manh	contest	0.001	0.4	-	18.95	1
100	5000-1.bin	manh	contest	0.001	0.4	-	188.42	0.99

Table 2: Δεύτερο Παραδοτέο

k	filename	metric	data_type	delta	sampling rate	rpt_init	CPU_time	Recall
100	1000-1.bin	manh	contest	0.001	0.4	yes	12.83	1
100	1000-1.bin	eucl_opt	contest	0.001	0.4	yes	13.90	1
100	1000-1.bin	eucl	contest	0.001	0.4	yes	14.29	0.98
100	5000-1.bin	manh	contest	0.001	0.4	yes	120.76	0.99

Table 3: Τρίτο Παραδοτέο

6.2. Βέλτιστες Δομές

Η βέλτιστη δομή για το τελικό παραδοτέο είναι τα avl trees και τα heaps και για τον υπολογισμό του brute force χρησιμοποιήθηκαν min_heaps.

7. Βιβλιογραφία

References

[1] Kai Li Wei Dong, Moses Charikar. Efficient k-nearest neighbor graph construction for generic similarity measures, March 2011.

Περιγραφή υλοποίησης του αλγορίθμου σε python: PyNNDescent
Dan Kluser and Jonas Bokstaller and Samuel Rutz and Tobias Buner. Fast Single-Core K-Nearest Neighbor Graph Computation
nndescent implementation in C++ for python