Κ23α - Ανάπτυξη Λογισμικού Για Πληροφοριακά Συστήματα

Χειμερινό Εξάμηνο 2024 2025

Άσκηση 1 – Παράδοση: Δευτέρα 4 Νοεμβρίου 2024

Αναζήτηση εγγύτερου γείτονα (Nearest Neighbor Search)

Η αναζήτηση εγγύτερου γείτονα (nearest neighbor search - NSS), είναι ένα πρόβλημα βελτιστοποίησης εύρεσης του σημείου σε ένα δεδομένο σύνολο που είναι εγγύτερα (ή πιο όμοιο) σε ένα δεδομένο σημείο. Η εγγύτητα συνήθως εκφράζεται ως μια μετρική ανομοιότητας (dissimilarity), δηλαδή όσο λιγότερο όμοια είναι τα αντικείμενα, τόσο μεγαλύτερη η τιμή της ανομοιότητας.

Τυπικά, το πρόβλημα εύρεσης εγγύτερου γείτονα ορίζεται ως εξής: έστω ένα σύνολο *S* σημείων σε ένα χώρο *M* και ένα σημείο επερώτησης $q \in M$. Βρείτε το σημείο στο *S* που είναι πιο κοντά στο *q*. Συνήθως ο *M* είναι μετρικός χώρος και η ανομοιότητα εκφράζεται ως μετρική απόστασης, που είναι ένα συμμετρικό μέγεθος και ικανοποιεί την τριγωνική ανισότητα. Επίσης μπορεί να είναι χώρος *d* διαστάσεων, όπου η ανομοιότητα μετράται χρησιμοποιώντας μεταξύ άλλων την Ευκλείδεια απόσταση μεταξύ των διανυσμάτων.

k-Nearest Neighbors (KNN)

Μια γενίκευση του προβλήματος αναζήτησης του εγγύτερου γείτονα είναι η αναζήτηση των *k* εγγύτερων γειτόνων, όπου χρειάζεται να βρεθούν τα *k* εγγύτερα σημεία (k-Nearest Neighbor Search - KNN). Μία συνήθης τεχνική είναι η χρήση του γράφου εγγύτερων γειτόνων (k-Nearest Neigbors Graph - KNNG). Ο KNNG γράφος για ένα σύνολο αντικειμένων V είναι ένας κατευθυντικός γράφος με ένα σύνολο κορυφών V και μία ακμή από κάθε $v \in V$ στα K πιο όμοια αντικείμενα στο V, δεδομένης μιας μετρικής ομοιότητας.

Η κατασκευή του KNNG μέσω brute-force τεχνικών έχει πολυπλοκότητα $O(n^2)$ και μπορεί πρακτικά να εφαρμοστεί μόνο για μικρά σύνολα δεδομένων. Επίσης, η εύρεση των k ακριβώς εγγύτερων γειτόνων δεν μπορεί να πραγματοποιηθεί με πολυπλοκότητα μικρότερη από $O(n)$.

Προσεγγιστική εύρεση KNN

Επειδή η αναζήτηση της ακριβούς λύσης στο πρόβλημα σε τεράστιους όγκους δεδομένων είναι απαγορευτική σε χρονικό κόστος, έχουν προταθεί και αναζητηθεί προσεγγιστικοί αλγόριθμοι που ανταλλάσσουν την απόλυτη ακρίβεια με σημαντικά καλύτερο χρόνο. Αναλυτικότερα ο αλγόριθμος αναζήτησης θα επιστρέψει ένα σημαντικό ποσοστό από τους εγγύτερους γείτονες (π.χ. 90-95%) σε ένα κλάσμα του χρόνου που χρειάζεται για την εύρεση της ακριβούς λύσης.
Θεωρήστε μία επερώτηση q στο σύνολο X, και την έξοδο του αλγορίθμου ως ένα σύνολο X από k υποψήφιους εγγύς γείτονες. Έστω ότι T είναι το σύνολο των k ακριβώς εγγύτερων γειτόνων (ground-truth) στην ερώτηση q ανάμεσα στα σημεία του εξεταζόμενου χώρου. Ορίζουμε ως ανάκληση (recall) k-recall@k από το X ως $\frac{X \cap T}{X}$ Ο στόχος του προσεγγιστικού αλγορίθμου εύρεσης εγγύτερων γειτόνων είναι η μεγιστοποίηση της ανάκλησης, ενώ ο χρόνος αναζήτησης των αποτελεσμάτων παραμένει όσο μικρότερος γίνεται. Η ανταλλαγή (tradeoff) μεταξύ ανάκλησης και χρονικής καθυστέρησης είναι προφανής σε αυτό το πρόβλημα.

Αλγόριθμος κατασκευής γράφου Vamana

Μεταξύ των δημοφιλέστερων τεχνικών για την προσεγγιστική εύρεση των k εγγύτερων γειτόνων είναι οι αλγόριθμοι βασισμένοι σε γράφο.

Γράφοι σχετικής γειτονίας και ο αλγόριθμος GreedySearch

Οι περισσότεροι αλγόριθμοι προσεγγιστικής εύρεσης εγγύτερων γειτόνων δουλεύουν ως εξής: κατά τη διάρκεια της δημιουργίας του ευρετηρίου (index), κατασκευάζουν ένα γράφο $G = (P, E)$, βασισμένοι στις γεωμετρικές ιδιότητες του συνόλου P. Κατά τη διάρκεια της αναζήτησης για ένα διάνυσμα x_q, η αναζήτηση χρησιμοποιεί ένα φυσικά άπληστο ή βέλτιστης πρώτης διάσχισης αλγόριθμο όπως ο Αλγόριθμος 1 στο G. Ξεκινώντας από ένα ορισμένο σημείο $s \in P$, διασχίζουν τον γράφο σταδιακά πιο κοντά στο x_q.

Αλγόριθμος 1: GreedySearch(s, x_q, k, L)
Data: Graph G with start node s, query x_q, result size k, search list size $L \geq k$
Result: Result set $\mathcal{L}$ containing k-approx NNs, and a set $\mathcal{V}$ containing all the visited nodes

initialize sets $\mathcal{L} \leftarrow \{s\}$ and $\mathcal{V} \leftarrow \emptyset$
while $\mathcal{L} \setminus \mathcal{V} \neq \emptyset$ **do**
    let $p^* \leftarrow \arg\min_{p \in \mathcal{L} \setminus \mathcal{V}} ||x_p - x_q||$
    update $\mathcal{L} \leftarrow \mathcal{L} \cup N_{out}(p^*)$ and $\mathcal{V} \leftarrow \mathcal{V} \cup \{p^*\}$
if $\mathcal{L} > L$ **then**
        update $\mathcal{L}$ to retain closest L points to x_q
return [closest k points from $\mathcal{L}$; $\mathcal{V}$]

Έχει πραγματοποιηθεί εκτενής έρευνα στην κατανόηση κατασκευής αραιών γράφων για τους οποίους ο αλγόριθμος GreedySearch(s, x_q, k, L) συγκλίνει γρήγορα στους (προσεγγιστικά) εγγύτερους γείτονες για κάθε ερώτηση. Μια ικανή συνθήκη για να συμβεί αυτό, τουλάχιστον όταν οι ερωτήσεις είναι κοντά στα σημεία του συνόλου, είναι ο λεγόμενος γράφος αραιής γειτονίας (Sparse Relative Neighborhood Graph) [1]. Σε αυτό το γράφο,, οι εξερχόμενοι γείτονες (γείτονες προς τους οποίους έχει ακμές μία κορυφή) κάθε σημείου p καθορίζονται ως εξής: αρχικοποιείται ένα σύνολο $S = P \setminus \{p\}$. Όσο $S \neq \emptyset$, προσθέτουμε μία κατευθυνόμενη ακμή από το p στο p^*, όπου p^* είναι το κοντινότερο σημείο του p από το S και αφαιρούμε από το S όλα τα σημεία p', τέτοια ώστε $d(p, p') > d(p^*, p')$. Έτσι, ο αλγόριθμος GreedySearch($s, x_p, 1, 1$) που ξεκινά σε οποιοδήποτε $s \in P$ θα συγκλίνει στο p για όλα τα σημεία $p \in P$.
Ενω η κατασκευή είναι κατ' αρχήν ιδεατή, είναι αδύνατη η κατασκευή τέτοιων γράφων ακόμη και για μέτρια μεγέθη, καθώς η πολυπλοκότητα του είναι της τάξης $O(n^2)$. Βασισμένοι σε αυτή τη διαίσθηση, δημιουργήθηκαν πρακτικότεροι αλγόριθμοι που προσεγγίζουν αρκετά ικανοποιητικά τους γράφους αραιής γειτονίας με μικρότερη πολυπλοκότητα. Παρ' όλα αυτά, δεν υπάρχει ιδιαίτερη ευελιξία στον έλεγχο της διαμέτρου και της πυκνότητας της εξόδου του γράφου από αυτούς τους αλγόριθμους.

Η διαδικασία Γερού Κλαδέματος (Robust Prunng)

Όπως αναφέρθηκε παραπάνω, οι γράφοι αραιής γειτονίας είναι όλοι καλοί υποψήφιοι για τη διαδικασία αναζήτησης GreedySearch, αλλά μπορεί η διάμετρος τους να είναι αρκετά μεγάλη. Η αναζήτηση σε τέτοιους γράφους θα προκαλούσε πολλές ακολουθιακές αναγνώσεις για την εύρεση των γειτόνων των επισκεπτομένων κορυφών στο μονοπάτι αναζήτησης του Αλγορίθμου GreedySearch.

Για να μετριαστεί το πρόβλημα, προτείνεται να υπάρξει μείωση της απόστασης στην ερώτηση κατά ένα παράγοντα $a > 1$ σε κάθε κορυφή κατά μήκος του μονοπατιού αναζήτησης, αντί για απλή μείωση όπως στους γράφους αραιής γειτονίας. Θεωρήστε ότι ο κατευθυνόμενος γράφος, όπου οι εξερχόμενοι γείτονες κάθε σημείου p καθορίζονται από τη συνάρτηση RobustPrune($p, \mathcal{V}, a, R$) που παρουσιάζεται στον Αλγόριθμο 2. Σημειώστε ότι αν οι εξερχόμενοι γείτονες κάθε $p \in P$ καθορίζονται από τη RobustPrune($p, P \setminus \{p\}, a, n-1$), τότε η GreedySearch($s, p, 1, 1$), ξεκινώντας από κάθε s, θα συγκλίνει στο $p \in P$ σε λογαριθμικό αριθμό βημάτων, αν $a > 1$. Ακόμη και με αυτόν τον τρόπο, ο χρόνος που θα απαιτούνταν θα ήταν της τάξης του $O(n^2)$ για τη δημιουργία του ευρετηρίου. Για να μετριαστεί αυτό, ο αλγόριθμος Vamana καλεί τη διαδικασία RobustPrune($p, \mathcal{V}, a, R$) για ένα προσεκτικά επιλεγμένο $\mathcal{V}$ με πολύ λιγότερα από $n-1$ σημεία για να βελτιώσει τον χρόνο κατασκευής του ευρετηρίου.

Algorithm 2: RobustPrune($p, \mathcal{V}, a, R$)
Data: Graph G, point $p \in P$, candidate set $\mathcal{V}$, distance threshold $a \geq 1$, degree bound R
Result: G is modified by setting at most R new out-neighbors for p

$\mathcal{V} \leftarrow (\mathcal{V} \cup N_{out}(p)) \setminus \{p\}$
$N_{out}(p) \leftarrow \emptyset$
while $\mathcal{V} \neq \emptyset$ **do**
    $p^* \leftarrow \arg\min_{p' \in \mathcal{V}} d(p, p')$
    $N_{out}(p) \leftarrow N_{out}(p) \cup \{p^*\}$

    if $|N_{out}(p)| = R$ **then**
        break
    for $p' \in \mathcal{V}$ **do**
        if $a \cdot d(p^*, p') \leq d(p, p')$ **then**
            remove p' from $\mathcal{V}$

Ο Αλγόριθμος δημιουργίας ευρετηρίου Vamana

Ο αλγόριθμος Vamana κατασκευάζει έναν κατευθυνόμενο γράφο με μια επαναληπτική διαδικασία. Ο γράφος G αρχικοποιείται ώστε κάθε κορυφή να έχει R τυχαία επιλεγμένους εξερχόμενους γείτονες. Σημειώστε ότι ενώ ο γράφος είναι καλά συνδεδεμένος (well connected) όταν $R > logn$, οι τυχαίες συνδέσεις δεν εγγυώνται σύγκλιση του αλγορίθμου GreedySearch σε καλά αποτελέσματα. Έστω s το ενδιάμεσο (medoid) του συνόλου P, το οποίο θα είναι και το αρχικό σημείο του αλγορίθμου αναζήτησης. Ο αλγόριθμος στη συνέχεια επαναλαμβάνεται σε όλα τα σημεία $p \in P$ με τυχαία σειρά, και σε κάθε βήμα, ενημερώνει τον γράφο, καθιστώντας τον καταλληλότερο για να συγκλίνει στο p η συνάρτηση GreedySearch($s, x_p, 1, L$). Πράγματι, στην επανάληψη που αντιστοιχεί στο σημείο p, ο αλγόριθμος Vamana πρώτα εκτελεί τη συνάρτηση GreedySearch($s, x_p, 1, L$) στον τρέχοντα γράφο G και θέτει V_p ως το σύνολο των σημείων που έχει ήδη επισκεφθεί η συνάρτηση. Στη συνέχεια ο αλγόριθμος ενημερώνει το G εκτελώντας τη συνάρτηση RobustPrune($p, \mathcal{V}_p, a, R$) για να καθορίσει νέους εξερχόμενους γείτονες. Έπειτα, ο Vamana ενημερώνει τον γράφο G προσθέτοντας εισερχόμενες ακμές (p', p) για κάθε $p' \in N_{out}(p)$. Αυτό εξασφαλίζει ότι θα υπάρχουν συνδέσεις μεταξύ των κορυφών που έχουν ήδη επισκεφθεί στο μονοπάτι αναζήτησης και το p, επομένως εξασφαλίζει ότι ο ενημερωμένος γράφος θα ταιριάζει καλύτερα με τη GreedySearch($s, x_p, 1, L$), ώστε να συγκλίνει στο p.

Όμως, η πρόσθεση αντίστροφων ακμών της μορφής (p', p) μπορεί να οδηγήσει σε παραβίαση του βαθμού του p' και συνεπώς, οποτεδήποτε μία κορυφή p' παρουσιάσει εξερχόμενο βαθμό μεγαλύτερο από το κατώφλι R, θα πρέπει να μεταβληθεί ο γράφος, εκτελώντας τη συνάρτηση RobustPrune(

$p', N_{out}(p'), a, R$), όπου $N_{out}(p)$ είναι το σύνολο των εξερχόμενων γειτόνων του p'. Καθώς προχωράει ο αλγόριθμος, ο γράφος γίνεται καλύτερος και ταχύτερος για την GreedySearch.

Algorithm 3: Vamana Indexing algorithm
Data: Database P with n points where i-th point has coords x_i, parameters , L, R
Result: Directed graph G over P with out-degree <=R

```
initialize G to a random R-regular directed graph
let s denote the medoid of dataset P
let σ denote a random permutation of 1..n
for 1 ≤ i ≤ n do
    let [L; V] ← GreedySearch(s, x_σ(i), 1, L)
    run RobustPrune(σ(i), V, a, R) to update out-neighbors of σ(i)
    for all points j in N_out(σ(i)) do
        if |N_out(j) ∪ {σ(i)}| > R then
            run RobustPrune(j, N_out(j) ∪ {σ(i)}, a, R) to update
        out-neighbors of j
        else
            update N_out(j) ← N_out(j) ∪ σ(i)
```

Προδιαγραφές κώδικα

Ο κώδικας που θα υλοποιηθεί θα πρέπει να είναι μία βιβλιοθήκη που να παρέχει τις εξής δυνατότητες

- χειρισμός δεδομένων αυθαίρετου αριθμού διαστάσεων
- χειρισμός δεδομένων αυθαίρετου αριθμού αντικειμένων
- δυνατότητα εύρεσης των k εγγύτερων γειτόνων για ένα ή για όλα τα μέλη του συνόλου

Παράδοση εργασίας

Η εργασία είναι ομαδική, **2 ή 3 ατόμων**.
Γλώσσα υλοποίησης: C / C++
Περιβάλλον υλοποίησης: Linux (gcc > 9.4+).

Παραδοτέα: Η παράδοση της εργασίας θα γίνει με βάση το τελευταιο commit πριν την προθεσμία υποβολής στο git repository σας. **Η χρήση git είναι υποχρεωτική**.
Στο αρχείο README.md θα αναφέρονται τα εξής:

- Ονοματεπώνυμα και ΑΜ των μελών της ομάδας
- Αναφορά στο ποιο μέλος της ομάδας ασχολήθηκε με ποιο αντικείμενο

Επιπλέον, εκτός από τον πηγαίο κώδικα, θα παραδώσετε μια σύντομη αναφορά, με τις σχεδιαστικές σας επιλογές καθώς και να εφαρμόσετε ελέγχους ως προς την ορθότητα του λογισμικού με τη χρήση ανάλογων βιβλιοθηκών (Software testing). Η ορθότητα τυχόν μεταβολών θα ελέγχεται με αυτοματοποιημένο τρόπο σε κάθε commit/push μέσω github actions.

Datasets στα οποία μπορείτε να πειραματιστείτε: http://corpus-texmex.irisa.fr/

Αναφορές

1. Sunil Arya and David M. Mount. Approximate nearest neighbor queries in fixed dimensions. In Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA ’93, pages 271–280, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics. ISBN 0-89871-313-7. URL http://dl.acm.org/citation.cfm?id=313559.313768
2. Suhas Jayaram Subramanya, Devvrit, Rohan Kadekodi, Ravishankar Krishaswamy, and Harsha Vardhan Simhadri. 2019. DiskANN: fast accurate billion-point nearest neighbor search on a single node. Proceedings of the 33rd International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook, NY, USA, Article 1233, 13766–13776. URL https://dl.acm.org/doi/abs/10.5555/3454287.3455520