ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Κ23Γ: Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα – Χειμερινό ΄23

1η Εργασία: Αναζήτηση και συσταδοποίηση διανυσμάτων στη C/C++

Ημερ. Ανακοίνωσης: 6/10
Ημερ. Υποβολής: 29/11 – ΄Ωρα 23:59

Χατζησπύρου Μιχαήλ ΑΜ: 1115202000212
Καζάκος Παναγιώτης ΑΜ: 1115201900067

Περιεχόμενα

Γενική Περιγραφή

Κύριος στόχος της εργασίας μας είναι η ανάπτυξη λογισμικού για την αποδοτική αναζήτηση όμοιων διανυσμάτων μεγάλων διαστάσεων και την συσταδοποίησή τους. Ασχοληθήκαμε με την ανάπτυξη τριών διαφροετικων προγραμμάτων υλοποιώντας τους αλγορίθμους lsh (locality sensitive hashing), random projection to hypercube, καθώς και clustering με ανάθεση Lloyd's, Reverse lsh και Reverse hypercube. Τα προγράμματα είναι συμβατά με σύνολα δεδομένων τύπου MNIST τα οποία περιέχουν εικόνες χειρόγραφων ψηφίων σε μορφή διανυσμάτων με κάθε διάσταση να αναπαριστά ένα pixel. Οι αλγόριθμοι lsh και hypercube χρησιμοποιούνται στην αναζήτηση όμοιων διανυσμάτων. Πιο συγκεκριμένα υπολογίζονται οι αποστάσεις των διανυσμάτων κατά προσέγγιση αξιοποιώντας τον κατακερματισμό των δεδομένων. Με τις αποστάσεις καθορίζονται οι n κοντινότεροι γείτονες για κάθε εικόνα του συνόλου αναζήτησης καθώς και οι εικόνες που βρίσκονται εντός μιας ακτίνας. Για την συσταδοποιήση χρησιμοποιούμε την αρχικοποίηση kmeans++ που επιλέγει τυχαία διάσπαρτα αρχικά κέντρα και στην συνέχεια οι συστάδες καθορίζονται με βάση την επίλογη του χρήστη από τα ορίσματα, είτε με την ανάθεση Lloyd's, τον Reverses Lsh η Reverse Hypercube.

Οδηγίες Μεταγλώττισης

Έχει δημιουργηθεί ένα Makefile για την διευκόλυνση της μεταγλώττισης και την εκτέλεση των προγραμμάτων με κάποιες προκαθορισμένες παραμέτρους. Η μεταγλώττιση όλων των αρχείων και γίνεται με την εντολή make. Χρησιμοποιείται η τεχνική του separate compilation. Επίσης δίνεται η δυνατότητα στον χρήστη να μεταγλωττίσει μόνο το πρόγραμμα που επιθυμεί με τις παρακάτω εντολές:

- make lsh – για την μεταγλώττιση του προγράμματος Lsh
- make cube – για την μεταγλώττιση του προγράμματος Hypercube
- make cluster – για την μεταγλώττιση του προγράμματος clustering

Οδηγίες Χρήσης του Προγράμματος

Αφού ο χρήστης δημιουργήσει τα εκτελέσιμα αρχεία με όποια μέθοδο επιθυμεί μπορεί να τα εκτελέσει με την βοήθεια του Makefile χρησιμοποιώντας τις εξής εντολές:

- make run-lsh – για την εκτέλεση του προγράμματος Lsh με τα ορίσματα που δηλώνονται στην μεταβλητή ARGS_LSH

- make run-cube – για την εκτέλεση του προγράμματος Hypercube με τα ορίσματα που δηλώνονται στην μεταβλητή ARGS_CUBE
- make run-cluster – για την εκτέλεση του προγράμματος Cluster με τα ορίσματα που δηλώνονται στην μεταβλητή ARGS_CLUSTER

Οι μεταβλητές ARGS_LSH/CUBE/CLUSTER είναι δηλωμένες εντός του Makefile και η κάθε μια παίρνει τα κατάλληλα ορίσματα προκειμένου να εκτελεστεί το αντίστοιχο εκτελέσιμο αρχείο. Επιπλέον δίνεται στον χρήστη η δυνατότητα εκτέλεσης των προγραμμάτων μέσω terminal ως εξής:

- ./bin/lsh_main -d <input file> -q <query file> -k <int>? -L <int>? -o <output file> -N <number of nearest>? -R <radius>?

  Default arguments: k = 4, L = 5, N = 1, R = 1000

- ./bin/cube_main -d <input file> -q <query file> -k <int>? -M <int>? -probes <int> ? -o <output file> -N <number of nearest>? -R <radius>?

  Default arguments: k = 14, M = 10, probes = 2, N = 1, R = 1000

- ./bin/cluster_main -i <input file> -c <configuration file> -o <output file> -complete -m <method: Classic OR LSH or Hypercube>

  Το configuration file είναι απαραίτητο για να οριστούν τα ορίσματα των αλγορίθμων LSH και Hypercube και είναι στην εξής μορφή:
  1. number_of_clusters: <int> K of KMeans++
  2. number_of_vector_hash_tables: <int>? default: L=3
  3. number_of_vector_hash_functions: <int>? k of LSH for vectors, default: 4
  4. max_number_M_hypercube: <int>? M of Hypercube, default: 10
  5. number_of_hypercube_dimensions: <int>? k of Hypercube, default: 3
  6. number_of_probes: <int>? probes of Hypercube, default: 2

Όποια ορίσματα έχουν ερωτηματικό στο τέλος τους είναι προαιρετικά, ο αλγόριθμος θα χρησιμοποιήσει τα default.

Δομή Φακέλων Κώδικα

├── bin	Εκτέλεσιμα αρχεία από main συναρτήσεις του src και tests
├── build	Αντικειμενικά αρχεία που δεν έχουν συνδεθεί
├── conf	Περιέχει αρχείο ρυθμίσεων για το cluster πρόγραμμα
├── datasets	Εδώ τοποθετούνται τα σύνολα δεδομένων, εισόδου και αναζήτησης. Παραλείπονται στο παραδοτέο.
├── modules	Κομμάτια κώδικα οργανωμένα σε επιμέρους φακέλους, κλάσεις και συναρτήσεις που χρησιμοποιούνται στα προγράμματα για καλύτερη οργάνωση του κώδικα και επανα χρησιμοποιήσή του.
│ ├── Cluster	Υλοποίηση του clustering
│ ├── Common	Φάκελος που περιέχει κοινές δομές/υλοποιήσεις από όλα τα προγράμματα
│ │ ├── BruteForce	Υλοποίηση της εξαντλητικής αναζήτησης
│ │ ├── FileParser	Υλοποίηση του parser που διαβάζει τα αρχεία εικόνων
│ │ ├── HashFunction	Υλοποίηση της hash function h
│ │ ├── ImageDistance	Υλοποίηση γενικευμένης απόστασης
│ │ └── Utils	Διάφορα utils που χρησιμοποιούνται από όλους τους αλγόριμους
│ ├── Cube	Υλοποίηση του κύβου
│ └── Lsh	Υλοποίηση του Lsh
│ └── HashTable	Υλοποίηση των Hash tables για των αλγόριθμο Lsh
└── src	Περιέχει τις main συναρτήσεις που υλοποιούν τα τρία προγράμματα που ζητούνται

Φάκελος Modules

Περιέχει διάφορες συναρτήσεις και κλάσεις που ομαδοποιούν σημαντικές λειτουργίες των προγραμμάτων. Κάθε φάκελος έχει όνομα που ταιριάζει στα τρία διαφορετικά προγράμματα εκτός από τα Common modules που χρησιμοποιούνται σε όλα.

Φάκελος SRC

Περιέχει τις τρείς διαφορετικές main συναρτήσεις για τα προγράμματα lsh, cube και cluster. Οι main συναρτήσεις είναι υπεύθυνες να λύσουν το ζητούμενο πρόβλημα κάθε φορά με αφηρημένο τρόπο διαβάζοντας τα κατάλληλα αρχεία, συνδυάζοντας τα απαραίτητα modules και εκτυπώνοντας τα αποτελέσματα.

Σημαντικές Δομές, Κλάσεις και Συναρτήσεις

Image: Αναπαριστά ένα σημείο από ενα MNIST σύνολο δεδομένων αποθηκεύοντας ένα αναγνωριστικό (id) και το διάνυσμα (pixels).

Neighbor: Αναπαριστά έναν γείτονα ενός σημείου. Αποθηκεύει το ίδιο το σημείο (image) αλλά και την απόσταση (distance) από το σημείο για το οποίο είναι γείτονας.

DistanceMetric: Enum τύπος που περιέχει τις Manhattan και Ευκλείδια μετρικές.

ImageDistance: Αποθηκεύει ποιά μετρική θα χρησιμοποιηθεί για το κάθε πρόγραμμα. Η αρχικοποίηση γίνεται μία φορα σε κάθε main και μπορεί να χρησιμοποιηθεί δυναμικά οπουδήποτε έχει κληθεί όταν τρέχει η εκάστοτε main.

CmdArgs: Αναλύει τα ορίσματα κάθε main συνάρτησης και τα αποθηκεύει σε μία εύκολα προσβάσιμη δομή. Για κάθε main υπάρχει ξεχωριστή κλάση (LshCmdArgs, CubeCmdArgs, ClusterCmdArgs).

FileParser: Αναλύει τα αρχεία εισόδου και αν ταιριάζουν στη μορφή ενός MNIST συνόλου δεδομένων, τότε αποθηκεύει τα μεταδεδομένα και τις εικόνες στην προσωρινή μνήμη του προγραμμάτος.

Utils: Περιέχει διάφορες σύντομες συναρτήσεις που χρησιμοποιούνται σε όλα τα προγράμματα.

BruteForce: Πραγματοποιεί αναζήτηση K πραγματικών πλησιέστερων γειτόνων ελέγχοντας για ένα σημείο αναζήτησης την απόσταση του με όλα τα σημεία.

HashFunction: Αποθηκεύει τις διάφορες παραμέτρους μιας hash συνάρτησης. Ο υπολογισμός της τιμής του κατακερματισμού γίνεται με τη μέθοδο hash. Στη μέθοδο αυτή χρησιμοποιείται ο $h(p) = \left\lfloor \frac{pv+t}{w} \right\rfloor \in Z$ από τις διαφάνειες.

AmpLsh: Δημιουργεί και αποθηκεύει τις ρυθμίσεις για μία amplified hash συνάρτηση συνδυάζοντας k διαφορετικές βασικές HashFunction. Η τιμή hash υπολογίζεται με τη μέθοδο hash αξιοποιώντας τον τύπο $g(p) = \sum_1^k r_i h_i(p) \ mod \ M$ από τις διαφάνειες, με M τον πρώτο $2^{32} - 5$.

HashTable: Πίνακας από διαφορετικά buckets (συνήθως n/8) στα οποία αποθηκεύονται τα διανύσματα. Χρησιμοποιείται στον αλγόριθμο lsh ο οποίος δημιουργεί L διαφορετικά hash tables. Το HashTable χρησιμοποιεί την AmpLsh δηλαδή $g(p)\ mod\ TableSize$.

LSH

Ο LSH επιλέγει μια οικογένεια συναρτήσεων κατακερματισμού που έχουν σχεδιαστεί για να διασφαλίζουν ότι παρόμοια σημεία δεδομένων είναι πιο πιθανό να κατακερματιστούν στον ίδιο κάδο. Τα σημεία δεδομένων κατακερματίζονται χρησιμοποιώντας αυτές τις συναρτήσεις, δημιουργώντας κουβάδες που ομαδοποιούν παρόμοια δεδομένα. Τέλος το Lsh εκτελεί αυτή την διαδικασία για πολλά hashtables.

Approximate Nearest Neighbors

Κατά την αναζήτηση των K-NN ο αλγόριθμος Lsh βρίσκει τον κουβά του query και ψάχνει για τους κοντινότερους γείτονες αποθηκεύοντας τους σε μια δομή συνόλου. Προκειμένου να γίνει εξοικονόμηση μνήμης ο αλγόριθμος αποθηκεύει μόνο τους knn οπότε σε περίπτωση που το σύνολο ξεπεράσει τον αριθμό k των κοντινότερων γειτόνων βγάζει το στοιχείο με την μεγαλύτερη απόσταση. Το σύνολο επιλέχθηκε για να αποφεύγονται οι διπλότυπες εγγραφές αφού λόγω των πολλαπλών hash tables που διαθέτει ο αλγόριθμος είναι πιθανό ένα σημείο να βρεθεί σε παραπάνω από ένα query bucket με αποτέλεσμα να μετρηθεί παραπάνω φορές.

Range Search

Για την αναζήτηση εντός περιοχής, χρησιμοποιείται πάλι σύνολο για τους λόγους που προαναφέρθηκαν πριν. Η βασική διαφορά είναι ότι το μόνο που ελέγχει ο αλγόριθμος είναι η απόσταση του σημείο από το query να είναι μικρότερη ή ίση της ακτίνας.

Cube

Σκοπός ειναι η μείωση των πολλών διαστάσεων των δεδομένων στις διαστάσεις k του υπερκύβου. Αρχικά δημιουργούμε k τυχαία διανύσματα που χρησιμοποιούνται για την δημιουργία k συναρτήσεων κατακερματισμού. Εφαρμόζουμε όλες τις συναρτήσεις κατακερματισμού για κάθε εικόνα και αντιστοιχίζουμε την κάθε hash τιμή τυχαία σε 0 ή 1. Οι αντιστοιχισμένες αυτές τιμές από τις τιμές όλων των hash συναρτήσεων για κάθε εικόνα προστίθενται σε ένα string. Το αποτέλεσμα κάθε φορά είναι μία κορυφή του υπερκύβου όπου κάθεμια είναι και ένα bucket ενός hash table με εικόνες. Επομένως οι εικόνες μπορούν να αναζητηθούν εύκολα σε λιγότερες διαστάσεις.

Approximate Nearest Neighbors

Κατά την αναζήτηση των K-NN ο αλγόριθμος βρίσκει μέσω του δυαδικού συστήματος τον κουβά του query και ψάχνει για τους κοντινότερους γείτονες. Για την αποθήκευση των

σημείων χρησιμοποιήσαμε την δομή της ουράς προτεραιότητας. Στην συνέχεια ο αλγόριθμος βρίσκει όλους τους γειτονικούς κουβάδες που έχουν hamming distance 1, 2, …Η μέχρι να φτάσουμε τον μέγιστο αριθμό υποψήφιων σημείων ή τον μέγιστο αριθμό υποψήφιων κουβάδων προς έλεγχο.

Range Search

Για την αναζήτηση εντός περιοχής, χρησιμοποιείται ουρά προτεραιότητας ο αλγόριθμος ακολουθεί την ίδια διαδικασία με την διαφορά ότι ελέγχει αν η απόσταση βρίσκεται εντός της ακτίνας.

Cluster

Αρχικοποίηση KMeans++

Διαλέγουμε τυχαία ένα σημείο ως κεντροειδές για την πρώτη συστάδα από το σύνολο δεδομένων. Στην συνέχεια, σε κάθε μια από τις υπόλοιπες συστάδες, για κάθε σημείο του συνόλου δεδομένων, το οποίο δεν έχει επιλεχθεί ήδη ως κεντροειδές, βρίσκουμε την ελάχιστη απόστασή του από τα υπάρχοντα κεντροειδή. Απο αυτές τις ελάχιστες αποστάσεις βρίσκουμε τη μέγιστη την οποία την χρησιμοποιούμε για να κανονικοποιηθούν οι τιμές των ελάχιστων αποστάσεων. Υπολογίζουμε τα μερικά αθροίσματα με τον τύπο $\sum_1^r \frac{D_i}{\text{maxD}}^2$ από τις διαφάνειες, όπου r ο αριθμός των σημείων πλην τον αριθμό των επιλεγμένων κεντροειδών. Στην συνέχεια επιλέγεται ένας τυχαίος αριθμός x με ομοιόμορφη κατανομή στο διάστημα 0 έως την τελευταία τιμή του πίνακα μερικών αθροισμάτων άρα και του μεγαλύτερου. Το επόμενο κεντροειδές που επιλέγεται για την συστάδα που επεξεργαζόμαστε είναι η θέση του πίνακα μερικών αθροιμάτων όπου η τιμή του x είναι πιο κοντά, χρησιμοποιώντας δυαδική αναζήτηση.

Lloyd’s

Η μέθοδος Lloyd’s υλοποιείται παίρνοντας ένα σύνολο από initialized clusters μέσω της μεθόδου KMeans++. Στην συνέχεια, ο αλγόριθμος, προσπελαύνει όλες τις δοσμένες εικόνες και για κάθε μια από αυτές υπολογίζει την απόσταση της από τα κέντρα των cluster. Με την βοήθεια ενός unordered_set ο αλγόριθμος γνωρίζει αν η εικόνα είχε ανατεθεί, προηγουμένως, σε διαφορετικό cluster και κάνει τις απαραίτητες ανανεώσεις κέντρων. Η διαδικασία τερματίζεται όταν δεν υπάρξει καμία νέα ανάθεση.

Reverse Range Search Lsh / Hypercube

Η μέθοδο Reverse Range Search υλοποιείται παίρνοντας ένα σύνολο από initialized clusters μέσω της μεθόδου KMeans++. Ο αλγόριθμος εκτελεί επαναληπτικά range search, είτε με την μέθοδο lsh είτε με την μέθοδο hypercube για τα centroids, αναθέτει τα σημεία, ανανεώνει τα κέντρα και προσαρμόζοντας την ακτίνα αναζήτησης. Η διαδικασία

συνεχίζεται μέχρι τα range searches να μην δίνουν νέα σημεία. Όταν ένα σημείο εμπίπτει σε πολλαπλές ακτίνες αναζήτησες, εκχωρείται στο πλησιέστερο κέντρο με βάση τις αποστάσεις. Τέλος, τα μη εκχωρημένα σημεία αναθέτονται με την μέθοδο Lloyd's όπως αυτή αναλύθηκε παραπάνω.

Hyperparameters

Ύστερα από πειραματισμούς καταλήξαμε στις εξής υπερπαραμέτρους:

- Lsh – k = 4, L = 5, w = 2240
- Hypercube – k = 14, M = 6000, probes = 15, w = 2240

  Επιλέξαμε το w = 2240, γιατί μπορούμε να υποθέσουμε αυθαίρετα την προσεγγιστική μέση απόσταση των pixels των εικονων σε κάθε διάσταση να είναι ίση με 80. Άρα θα έχουμε τη μέση απόσταση των εικόνων ίση με $\sqrt{\sum_0^{783}(x_i - y_i)^2} = \sqrt{80^2 784} = 2240$.

- Cluster
  1. Lsh – τα ίδια με παραπάνω εκτός από το w = 10
  2. Cube – k = 3, M = 4000, probes = 15, w = 10

Αναφορές

Locality-sensitive hashing

Locality-Sensitive Hashing Scheme Based on p-Stable Distributions

Similarity Search in High Dimensions via Hashing

Proximity problems for high-dimensional data

K-Means++

3 versions of K-Means

Hamming Distance

Incremental averaging

High resolution clock