Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα
Εργασία 2

Αθανασία Τουρνάκη - ΑΜ: 1115201600172
Δημήτριος Μυλωνόπουλος - ΑΜ: 1115201600112

Στην παρούσα εργασία μας ζητήθηκε να εφαρμόσουμε με διάφορες μεθόδους τον αλγόριθμο k-means για συσταδοποίηση σημείων και καμπυλών.

Πιο συγκεκριμένα, υλοποιήθηκαν συναρτήσεις αρχικοποίησης των clusters, ανάθεσης στοιχείων σε clusters και ενημέρωσης των κεντροειδών κάθε cluster μετά από κάθε επανάληψη. Για κάθε στάδιο του clustering έπρεπε να υλοποιηθούν δύο διαφορετικές μέθοδοι. Επιπλέον, οι μέθοδοι, που επιλέγονται από τον χρήστη για κάθε στάδιο, ήταν απαραίτητο να μπορούν να συνδυαστούν, ώστε να επιτευχθεί η συσταδοποίηση.

Αναλυτικά, για κάθε στάδιο, οι μέθοδοι που υλοποιήθηκαν:

- **Αρχικοποίηση:**
    1) Simple Initialization
    2) K-means++
- **Ανάθεση:**
    1) Lloyd's Algorithm
    2) Range-Search using LSH
- **Ενημέρωση:**
    1) Partitioning Around Medoids (PAM)
    2) Σημεία: Mean Vector, Καμπύλες: DTW Barycenter Averaging (DBA)

~~~~~~

Περιγραφή Αρχείων:

Clustering.cpp: Περιέχει την υλοποίηση της template κλάσης clustering η οποία περιέχει συναρτήσεις για όλες τις μεθόδους που αναφέρονται παραπάνω, καθώς επίσης και για το silhouette evaluation που ζητήθηκε. Λίγα σχόλια για τις συναρτήσεις:

- **initializePlus():** Στη συνάρτηση αυτή υλοποιήθηκε ο αλγόριθμος k-means++ με τον τρόπο που υποδεικνύεται στις διαφάνειες. Για την αναζήτηση των σημείων σε κάθε βήμα επιλογής των κεντροειδών του αλγορίθμου, χρησιμοποιείται δυαδική αναζήτηση, ώστε αυτά να επιλέγονται σε λογαριθμικό χρόνο. Η γεννήτρια τυχαίων αριθμών ακολουθεί ομοιόμορφη κατανομή.
~~~~~~

- **initialize():** Στη συνάρτηση αυτή επιλέγονται με ομοιόμορφη κατανομή τυχαία στοιχεία από το dataset ως κεντροειδή ανάλογα με το πόσα clusters υπάρχουν.
- **lloydAssign():** Στη συνάρτηση αυτή, για κάθε στοιχείου του dataset που δεν είναι κεντροειδές υπολογίζεται η απόσταση του από όλα τα κεντροειδή και αυτό ανατίθεται στο cluster του κεντροειδούς από το οποίο απέχει την μικρότερη απόσταση.
- **rangeSearchAssign():** Στη συνάρτηση αυτή χρησιμοποιείται δομή LSH με σκοπό την επιτάχυνση της διαδικασίας ανάθεσης των στοιχείων του dataset στα clusters. Τα στοιχεία που βρίσκονται στο ίδιο bucket με ένα κεντροειδές ανατίθενται στο cluster του. Αν ένα σημείο ανήκει λόγω πολλαπλών hash tables σε παραπάνω από ένα κεντροειδές, τότε ανατίθεται σε εκείνο που απέχει από αυτό τη μικρότερη απόσταση. Τα στοιχεία που δεν έχουν ανατεθεί σε κάποιο cluster μετά από αυτή τη διαδικασία ανατίθενται στα clusters με τη χρήση του αλγορίθμου του Lloyd που αναλύθηκε παραπάνω.
- **pamUpdate():** Στη συνάρτηση αυτή υπολογίζεται για κάθε σημείο(συμπεριλαμβανομένου και του κεντροειδούς) η αντικειμενική συνάρτηση, που σε αυτή αυτήν την περίπτωση είναι το άθροισμα των αποστάσεων του στοιχείου με όλα τα υπόλοιπα στοιχεία του cluster και επιλέγεται ως νέο κεντροειδές αυτό με την μικρότερη τιμή που επιστρέφει η αντικειμενική συνάρτηση.
- **meanVectorUpdate():** Στη συνάρτηση αυτή, ως νέο κεντροειδές επιλέγεται το μέσο σημείο-διάνυσμα του cluster, δηλαδή αυτό που η κάθε διάστασή του είναι η μέση τιμή της ίδιας διάστασης όλων των σημείων του cluster. Για να αναπαρασταθεί αυτό το διάνυσμα, που προφανώς δεν ανήκει στο dataset, δημιουργείται ένα νέο instance της κλάσης Point που σημειώνεται ως fake για να ξεχωρίζει από το σημεία του dataset.
- **dbaUpdate():** Στη συνάρτηση αυτή, για κάθε cluster υπολογίζεται το μέσο μήκος καμπύλης(λ). Στη συνέχεια επιλέγεται τυχαία με ομοιόμορφη κατανομή μία τυχαία καμπύλη του cluster με μέγεθος μεγαλύτερο ή ίσο του λ. Από αυτή επιλέγεται τυχαία(με ομοιόμορφη κατανομή) μία αύξουσα υπακολουθία μήκους λ, έστω seqC. Έπειτα, μέχρι η υπακολουθία να μην αλλάζει σε μεγάλο βαθμό ή να ολοκληρωθεί ένας συγκεκριμένος αριθμός επαναλήψεων, ακολουθείται η εξής διαδικασία:
    1) Για κάθε σημείο του cluster υπολογίζεται μία βέλτιστη διάσχιση ανάμεσα σε αυτό και την seqC.
    2) Για κάθε σημείο της ακολουθίας seqC υπολογίζεται η μέση τιμή της τετμημένης και της τεταγμένης όλων των άλλων σημείων που αντιστοιχίζονται με αυτό στη διάσχιση που βρέθηκε παραπάνω. Αυτές θα είναι οι νέες συντεταγμένες του σημείου.

3) Δημιουργείται μία νέα “fake”(όπως παραπάνω) καμπύλη, η οποία αντικατοπτρίζει τις αλλαγές που έγιναν στο βήμα 2.
4) Υπολογίζεται η απόσταση μεταξύ της παλιάς και της ενημερωμένης ακολουθίας seqC για να γίνει έλεγχος σύγκλισης.

dataStructs.hpp: Το συγκεκριμένο αρχείο περιέχει τον ορισμό των κλάσεων για την αποθήκευση των σημείων και των καμπυλών. Ως σύμβαση έχουμε στις κλάσεις σημείων
(class Point) έναν δείκτη σε κλάσεις καμπυλών αν αυτά τα σημεία προκύψουν από τη δημιουργία διανυσμάτων grid Curves.

fileReading.cpp: Περιέχει την υλοποίηση της κλάσης Reading που χρησιμοποιείται για την ανάγνωση των αρχείων εισόδου και τη μετατροπή τους σε vector σημείων ή καμπυλών.

bruteForce.cpp: Περιέχει τη συνάρτηση calculateW() που βρίσκει το W για την υλοποίηση του αλγορίθμου lsh μέσω του υπολογισμού της μέσης
απόστασης των σημείων ενός δείγματος του dataset με τους πλησιέστερους γείτονές τους.

objectiveFunctions.cpp: Περιέχει τις αντικειμενικές συναρτήσεις για σημεία και καμπύλες που επιστρέφουν τη μικρότερη απόσταση ενός στοιχείου από όλα τα υπόλοιπα που ανήκουν στο ίδιο cluster. Αυτές χρησιμοποιούνται στην υλοποίηση της pamUpdate().

dynamicTimeWarping.cpp: Υλοποίηση του αλγορίθμου σύγκρισης αποστάσεων καμπυλών DTW με χρήση δυναμικού προγραμματισμού.

metrics.cpp: Περιλαμβάνει τις μετρικές l1 και l2την εύρεση των αποστάσεων σημείων. (Manhattan και Ευκλείδεια) που ζητήθηκαν για την εύρεση των αποστάσεων σημείων.

hashTable.cpp: Το hashtable υλοποιήθηκε ως εξής:

- Για το lsh δημιουργείται hashtable που είναι ένας unordered_map, όπου το key είναι ένας δείκτης στο κεντροειδές και το entry που συνδέεται με αυτό είναι το cluster του συγκεκριμένου κεντροειδούς. Το hashtable περιλαμβάνει την hash function που έχει παρουσιαστεί στη τάξη, η οποία περιέχει ειδική συνάρτηση για την εξασφάλιση αποφυγής overflow στην ύψωση μεγάλων δυνάμεων.

LSH.cpp: Περιέχει την template κλάση LSH η οποία προσαρμόζεται ανάλογα με το αν θα εισαχθούν σε αυτό σημεία ή σημεία που προέρχονται από καμπύλες.
Η συνάρτηση του constructor αρχικοποιεί τα hash tables και εισάγει στα σε αυτα τα σημεία του Dataset.

gridCurve.cpp: Σε αυτό το αρχείο υλοποιήθηκε η διαδικασία προβολής σημείων σε grid με σκοπό τη δημιουργία διανυσμάτων. Χρησιμοποιήθηκε συνάρτηση που υπολογίζει την
παράμετρο δέλτα με βάση τη μέση απόσταση των διαδοχικών σημείων των καμπυλών του dataset. Για τον υπολογισμό του w χρησιμοποιήθηκε η συνάρτηση **calculateW()** του
bruteforce.cpp, εφόσον, ανάλογα με τον πίνακα-παράμετρο τ, οι μέσες αποστάσεις των διανυσμάτων που προήλθαν από καμπύλες ήταν διαφορετικές. Επιπλέον, αφαιρέθηκαν τα συνεχόμενα διπλότυπα ζεύγους συντεταγμένων από το παραχθέν διάνυσμα και χρησιμοποιήθηκε για padding η snapped τιμή της μέγιστης συντεταγμένης που βρέθηκε στο dataset. Κάθε διάνυσμα έχει διαστάσεις όσο το διπλάσιο του μέγιστου αριθμού σημείων σε καμπύλη του dataset.

main.cpp: Σε αυτό το αρχείο περιέχεται η αρχικοποίηση του προγράμματος με βάση το input του χρήστη από τη γραμμή εντολών, δηλαδή διαβάζει το αρχείο input και δημιουργεί ένα vector σημείων ή καμπυλών, διαβάζει το αρχείο cluster.conf που περιέχει δεδομένα για κάποιες μεθόδους και δημιουργεί το αρχειο output. Επίσης, περιέχει κώδικα για ένα υποτυπώδες UI στο terminal, το οποίο επιτρέπει στον χρήστη να διαλέξει ποιες μεθόδους θα χρησιμοποιηθούν για την εκτέλεση του clustering.

mainTest.cpp: Αυτό το αρχείο δημιουργήθηκε για σκοπούς unit Testing. Χρησιμοποιήθηκε η βιβλιοθήκη gtest. Εδώ γίνονται έλεγχοι σε κάποιες συναρτήσεις ανάγνωσης αρχείων καθώς και σε κάποιες συναρτησεις μέτρησης αποστάσεων από το **dynamicTimeWarping.cpp**.

~~~~~~

Σχετικά με τη σύγκλιση: Στο αρχείο main, επιτρέπεται να γίνουν μόνο έως και 30 επαναληψεις για ανάθεση και ενημέρωση στο cluster μέχρι να επιτευχθεί σύγκλιση. Πέρα από αυτό όμως, σε κάθε συνάρτηση update() της κλάσης clustering γίνεται έλεγχος για σύγκλιση του παλιού και του καινούριου κεντροειδούς κάθε cluster.

~~~~~~

Σύγκριση Αλγορίθμων

Μαζί με την εργασία έχει συμπεριληφθεί ένα αρχείο txt () το οποίο περιέχει εκτελέσεις όλων των συνδυασμών των αλγορίθμων που ζητήθηκαν προς υλοποίηση (Initialization, Assignment, Update) . Συγκεκριμένα υπάρχουν 8 εκτελέσεις για τα σημεία και άλλες 8 για τις καμπύλες. Από αυτές τις εκτελέσεις μπορούμε να εξάγουμε κάποια συμπεράσματα, για τα πλεονεκτήματα, τα μειονεκτήματα και τη χρησιμότητα αυτών των αλγορίθμων:

- **Initialization:** Παρατηρούμε ότι το Simple είναι χειρότερο από το Kmeans++. Δηλαδή χρειάζεται παραπάνω iterations για σύγκλιση, αλλά επίσης πέφτει πολύ πιο συχνά σε τοπικά ελάχιστα, τα οποία δεν είναι ολικά. Γι αυτό και στα σημεία για παράδειγμα βλέπουμε οτι το avg Sihlouette είναι γύρω στο 0.5 για Simple ενώ γύρω στο 0.9 για Kmeans++. Άρα καταλαβαίνουμε πως το παραπάνω κόστος του Initialization με Kmeans++ είναι πολύ μικρότερο από το κόστος που μπορεί να έχουν τα παραπάνω iterations, δηλαδή η πιο αργή σύγκλιση, αλλά και ο μικρότερος κίνδυνος για τοπικά ελάχιστα.

- **Assignment:** Με τη μέθοδο του Range Search παρατηρούμε αισθητή μείωση του χρόνου εκτέλεσης του αλγορίθμου στις περισσότερες περιπτώσεις. Αυτό προφανώς είναι άμεσα συνυφασμένο με το πόσα στοιχεία θα έχουν πέσει στο ίδιο bucket με τα κεντροειδή που επιλέγονται, αν είναι λίγα τότε προφανώς δε θα υπάρχει η ίδια επιτάχυνση. Η μείωση του χρόνου όμως έρχεται με μια ελαφριά μείωση της ακρίβειας των αποτελεσμάτων, επομένως ανάλογα με το DataSet και τις απαιτήσεις μας επιλέγουμε τον κατάλληλο αλγόριθμο για το Clustering.

- **Update:** Παρατηρούμε ότι το PAM επιφέρει καλύτερα αποτελέσματα από το Mean Vectors-DBA. Κάτι τέτοιο είναι αναμενόμενο μιας και ο αλγόριθμος εφαρμόζει μια brute Force μέθοδο για να βρει ποιό θα είναι το ιδανικό νέο κεντροειδές για το Cluster. Φυσικά αυτό έρχεται με ένα κόστος χρόνου και συχνά η μέθοδος του Mean βγάζει αντάξια αποτελέσματα, άρα μας εξυπηρετεί η ταχύτητα της στις περισσότερες περιπτώσεις clustering.

~~~~~~
~~~~~~

Οδηγίες Μεταγλώττισης:

Για την μεταγλώττιση του προγράμματος ανακατευθείνεστε στον φάκελο “src” του Project και εκτελείτε την εντολή “make”. Δεδομένου ότι χρησιμοποιείτε η google test βιβλιοθήκη το compilation θα γίνει αν ικανοποιούνται τα dependencies. Αν δεν εχετε την βιβλιοθήκη google test για ubuntu distros η εντολή είναι η εξής:

“sudo apt-get install libgtest-dev -y ; cd /usr/src/gtest ; sudo apt-get install cmake ; sudo cmake CMakeLists.txt; sudo make ; sudo cp *.a /usr/lib”

~~~~~~

Οδηγίες Εκτέλεσης:

Για την εκτέλεση του προγράμματος:

$./cluster –i <input file> –c <configuration file> -o <output file> -complete <optional>

Και μέσω του μενού του προγράμματος επιλέγετε τους αλγορίθμους που επιθυμείτε να εκτελέσετε.

Για την εκτέλεση του testing:
./mainTest

Για τον καθαρισμό του src, εκτελείτε την εντολή **make clean.**
~~~~~~