ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν
Πανεπιστήμιον Αθηνών
ΙΔΡΥΘΕΝ ΤΟ 1837

Εθνικο και Καποδιστριακο Πανεπιστημιο Αθηνων

K23: Αναπτυξη Λογισμικου για Πληροφοριακα Συστηματα

Τελική Αναφορά

Ομάδα:
Ιγγλέζου Μυρτώ
Νικολέτος Κωνσταντίνος

A.M.:
1115201700038
1115201700104

Χειμερινό εξάμηνο 2020-2021

Περιεχόμενα

1 Περιγραφή

1.1 Ζητούμενο

Στα sites ηλεκτρονικού εμπορίου, δεν υπάρχουν ακριβή πεδία για την καταχώρηση των προϊόντων. Συνεπώς, τα αναφερόμενα χαρακτηριστικά ίδιων προϊόντων διαφέρουν τόσο στα πεδία (keys) που καταγράφονται (αριθμός πεδίων, όνομα πεδίου), όσο και στις τιμές των πεδίων αυτών.

Ζητούμενο του συστήματος που υλοποιήσαμε είναι η αναγνώριση ίδιων προϊόντων που αντιστοιχούν σε διαφορετικές καταχωρήσεις. Στη βιβλιογραφία απαντάται ως αναγνώριση ταυτότητας (**entity resolution**) ή αποσαφήνιση (**disambiguation**).

1.2 Επισκόπηση λειτουργίας συστήματος

Στην αρχή εκτέλεσης του προγράμματος, διαβάζεται ο φάκελος που περιέχει τους φακέλους των specs και αποθηκεύονται οι πληροφορίες τους στις κατάλληλες δομές. Τότε, διαβάζεται το Dataset W και σχηματίζονται οι κλίκες και οι αρνητικές συσχετίσεις. Στη συνέχεια, δημιουργείται το λεξιλόγιο και υπολογίζονται τα διανύσματα για κάθε κάμερα.

Σχηματίζεται μετά, το σύνολο δεδομένων με ανακατεμμένες θετικές και αρνητικές συσχετίσεις και χωρίζεται σε τρία μέρη - train 60%, test 20%, validation 20%. Επείτα, σε κάθε επανάληψη γίνεται εκπαίδευση του μοντέλου με τον αλγόριθμο Mini-Batch GD του Logistic regression με χρήση πολυνηματισμού. Γίνεται εκτίμηση της εκπαίδευσης μέσω του test set και έπειτα επανεκπαιδεύεται το μοντέλο με όλα τα ζευγάρια του αρχικού Dataset που δεν γνωριζάμε κάποια συσχέτιση για αυτά.

Στο τέλος των επαναλήψεων γίνεται μια τελευταία εκτίμηση του μοντέλου μέσω του validation set.

2 Περιβάλλον υλοποίησης

- Γλώσσα προγραμματισμού: **C**
- Λειτουργικό σύστημα: **Ubuntu Linux 20.04** (σε Virtual Machine)
- Έκδοση gcc: **9.3.0**

3 Δεδομένα εισαγωγής στο σύστημα

Τα δεδομένα αποτελούνται από τα εξής αρχεία:

1. **Dataset X** (directory)
   Ένα σύνολο δεδομένων (dataset) που περιέχει περίπου 30000 προδιαγραφές προϊόντων (spec) σε μορφή JSON. Κάθε spec περιέχει μία λίστα από ζεύγη όνομα_ιδιότητας, τιμή_ιδιότητας που έχουν εξαχθεί από διαφορετική ιστοσελίδα και έχουν συλλεχθεί από 24 διαφορετικά web sites. Το σύνολο δεδομένων αυτό αναφέρεται ως .

   - Κάθε spec αποθηκεύεται ως ξεχωριστό αρχείο και τα αρχεία οργανώνονται σε καταλόγους (directories), όπου κάθε κατάλογος αντιστοιχεί σε διαφορετική πηγή δεδομένων (π.χ. www.alibaba.com, www.ebay.com)
   - Όλα τα specs αναφέρονται σε φωτογραφικές μηχανές και περιλαμβάνουν πληροφορίες για το μοντέλο της φωτογραφικής μηχανής (π.χ. Canon EOS 5D Mark II) και, πιθανώς, εξαρτήματα/αξεσουάρ (π.χ. σετ φακών, τσάντα, τρίποδα). Τα αξεσουάρ δεν συνεισφέρουν στην αναγνώριση του προϊόντος. Για παράδειγμα μία μηχανή Canon EOS 5D Mark II που πωλείται μαζί με μία τσάντα μεταφοράς θεωρείται το ίδιο προϊόν με μία Canon EOS 5D Mark II που πωλείται μόνη της

```
{
    "<page title>": "Samsung Smart",
    "brand": "Samsung",
    "dimension": "101 x 68 x 27.1 mm",
    "display": "LCD 3 Inches",
    "pixels": "Optical Sensor Resolution \n16.2 MP"
    "battery": "Li-Ion"
}
```

Γράφημα 1: Παράδειγμα εγγραφής σε μορφή JSON

2. **Dataset W** (sigmod_large_labelled_dataset.csv)
   Είναι ένα σημειωμένο dataset σε μορφή CSV, που περιέχει 3 στήλες "left_spec_id" ,"right_spec_id" και "label".

   - Το "spec_id" είναι μία μοναδική ταυτότητα για ένα spec και αποτελείται από το σχετικό μονοπάτι (path) του αρχείου spec. Σημειώστε ότι αντί για τον χαρακτήρα "", το spec_id χρησιμοποιεί τον ειδικό χαρακτήρα "" και ότι παραλείπεται η επέκταση ".json".
     Για παράδειγμα, το spec_id "www.ebay.com//1000" αναφέρεται στο αρχείο 1000.json, το οποίο βρίσκεται στον κατάλογο www.ebay.com. Όλα τα spec_id στο σημειωμένο dataset W αναφέρονται σε χαρακτηριστικά προϊόντων που περιλαμβάνονται στο dataset X

   - Κάθε γραμμή του σημειωμένου dataset αναπαριστά ένα ζεύγος spec. Αν η τιμή της στοιχείου label είναι 1 σημαίνει ότι το αριστερό και το δεξί spec αναφέρονται στο ίδιο προϊόν (δηλαδή ταιριάζουν). Η τιμή του στοιχείου label 0 σημαίνει ότι τα δύο specs αναφέρονται σε διαφορετικό προϊόν (δηλαδή ότι δεν ταιριάζουν).

```
1  left_spec_id, right_spec_id, label
2  www.ebay.com//1, www.ebay.com//2, 1
3  www.ebay.com//3, buy.net//10, 0
```

Γράφημα 2: Παράδειγμα γραμμής στο σημειωμένο dataset

3. **Dataset Υ** (sigmod_medium_labelled_dataset.csv)
   Αποτελεί ένα υποσύνολο του Dataset W.

4 Σχεδιαστικές επιλογές

4.1 Αποθήκευση αρχείων καμερών

Με το διάβασμα του Dataset X οι πληροφορίες που αφορούν κάθε κάμερα αποθηκεύονται σε ένα struct με την εξής δομή:

```
typedef struct CamSpec{
    char* name;
    int arrayPosition;
    int myClique;
    Set set;
    BF* bitArray;
    int* dictionaryWords;
    int * wordCounters;
    int numOfWords;
    DenseMatrix * DenseVector;

}CamSpec;
```

Γράφημα 3: Δομή αποθήκευσης πληροφοριών κάμερας

Για την γρήγορη αναζήτηση αυτών των πληροφοριών έχει δημιουργηθεί ένα hashtable, όπου κάθε θέση του δείχνει σε ένα Red Black Tree και κάθε κόμβος των δέντρων δείχνει σε ένα struct κάμερας. Επιπλέον, για να διατρεχθούν οι πληροφορίες αυτές υπάρχει ένας πίνακας που κάθε θέση του (arrayPosition) δείχνει επίσης σε ένα struct κάμερας. Το μέγεθος του πίνακα είναι ίσο με το πλήθος των json στο dataset X.

Κάθε κάμερα αποθηκεύει στην μεταβλητή dictionaryWords αριθμούς που αντιστοιχούν στις λέξεις του json. Επιπλέον, στον πίνακα wordCounters αποθηκεύονται οι φορές που εμφανίζεται η κάθε λέξη στην πρόταση, αριθμοί που χρειάζονται στον υπολογισμό του tf-idf.

4.2 Διανυσματοποίηση δεδομένων

Με το διάβασμα των αρχείων json, οι λέξεις που περιέχονται σε αυτά, περνάνε απο μια συνάρτηση επεξεργασίας κειμένου. Στη συνέχεια, αποθηκεύονται σε ένα ένα hashtable, όπου κάθε θέση του δείχνει σε ένα Red Black Tree. Κάθε λέξη αντιστοιχίζεται σε έναν μοναδικό αριθμό. Μετά από το διάβασμα όλων των αρχείων, βρίσκουμε το μέσο tf-ifd για κάθε λέξη και με την qsort επιλέγονται οι 1000 με το μεγαλύτερο μέσο. Έπειτα, σχηματίζεται ένας πίνακας για κάθε κάμερα, που κρατά τις μη μηδενικές τιμές του διανύσματος και την θέση τους.

4.3 Δομές δεδομένων

Στο σύνολο της εργασίας χρησιμοποιήθηκαν οι εξής δομές:

- **Λίστα**: Για τον σχηματισμό των κλικών υλοποιήθηκε ένας πίνακας από διπλά συνδεδεμένες λίστες, όπου κάθε κόμβος της λίστας έδειχνε σε μία κάμερα, που ανήκε στην κλίκα της συγκεκριμένης θέσης του πίνακα.

Μέθοδος	Καλύτερη	Χειρότερη
Εισαγωγή στην Αρχή	O(1)	O(1)
Εισαγωγή στο τέλος	O(1)	O(1)
Αναζήτηση	O(1)	O(n)

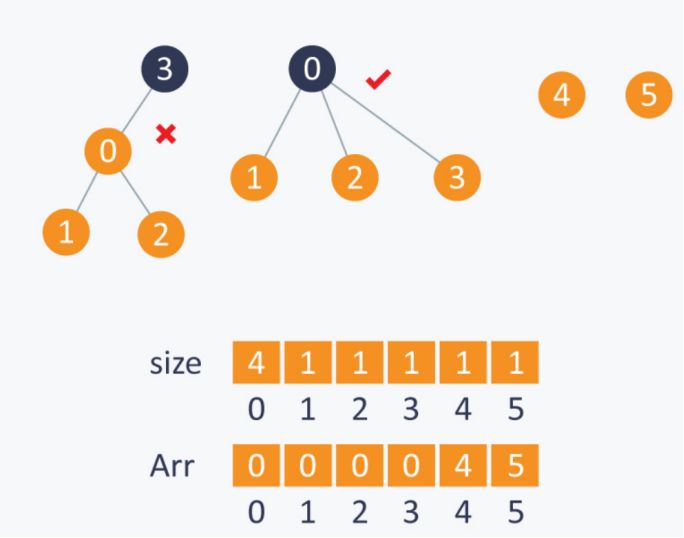
- **Hashtable-RBT**: Στις περιπτώσεις που χρειάστηκε να γίνει αναζήτηση δεδομένων, χρησιμοποιήθηκε Hashtable όπου κάθε θέση του οδηγούσε σε Red Black Tree. Οι κόμβοι κάθε δέντρου έδειχναν στην πληροφορία που ήταν προς αναζήτηση. Η δομή αυτή χρησιμοποιήθηκε στην περίπτωση του λεξιλογίου, όπου με κάθε νέα εισαγωγή λέξης έπρεπε να ελεγθεί αν η λέξη αυτή υπήρχε ήδη καταγεγραμμένη, και στην περίπτωση των specs, όπου πάλι έπρεπε να ελεγθεί αν κάποιο υπήρχε ήδη.

Μέθοδος	Καλύτερη	Χειρότερη
Εισαγωγή	O(1)*O(1)	O(1)*O(logn)
Αναζήτηση	O(1)*O(1)	O(1)*O(logn)

- **Bitarray**: Για να αποθηκευτούν οι σχέσεις μεταξύ κλικών αλλά και μεταξύ specs χρησιμοποιήθηκε Bitarray. Κάθε κλίκα έχει έναν πίνακα από bit, μεγέθους όσο είναι και το πλήθος των κλικών. Η κλίκα έχει αρνητική συσχέτιση με τις κλίκες που βρίσκονται στις θέσεις με τιμή 1. Ακόμα, στην υλοποιήση του τρίτου μέρους της εργασίας χρειάστηκε να γνωρίζουμε τα ζευγάρια που έχουν δημιουργηθεί, ώστε στην διαδικασία του retrain να μην εκπαιδεύονται τα ίδια. Αυτή η πληροφορία αποτυπώνεται, μέσω ενός bitArray σε κάθε spec μεγέθους ίσου με το πλήθος όλων των spec. Κάθε spec, αποθηκεύει στον πίνακα με 1 τις θέσεις που αντιστοιχούν σε specs που έχει είτε θετική, είτε αρνητική συσχέτιση.

Μέθοδος	Καλύτερη	Χειρότερη
Εισαγωγή	O(1)	O(1)
Αναζήτηση	O(1)	O(1)

- **Disjoint set**: Για την δημιουργία κλικών χρησιμοποιήθηκε η δομή δεδομένων Disjoint set. Είναι ουσιαστικά ένας πίνακας απο ακεραίους, όπου κάθε θέση του αντιπροσωπεύει ένα spec και η τιμή της θέσης αυτής αποθηκεύει τον "πατέρα" του spec, δηλαδή στοιχεία με ίδιο "πατέρα" ανήκουν και στην ίδια κλίκα. Με το διάβασμα του Dataset W, κάθε θετική συσχέτιση συνεπάγεται ότι τα δύο στοιχεία ανήκουν στην ίδια κλίκα. Τότε ο πίνακας ενημερώνεται.



Γράφημα 4: Παράδειγμα disjoint set

4.4 Πολυνηματισμός

Για την ταχύτερη εκτέλεση του προγράμματος έχει γίνει παραλλήλοποίηση σε δύο σημεία του προγράμματος. Πρίν όμως αναφερθεί η λειτουργία σε αυτά τα σημεία, θα γίνει μια συνοπτική εξήγηση της υλοποίησης του πολυνηματισμού. Στο σύστημα αυτή η λειτουργία είναι στοω φάκελο /modules/JobScheduler και αποτελείται από:

- Έναν χρονοπρογραμματιστή, που ουσιαστικά δέχεται δουλειές και αναλαμβάνει την ανάθεση τους σε νήματα, για προσωρινή αποθήκευση των εργασιών χρησιμοποιεί μια ουρά προτεραιότητας. Έχουμε μια Δεξαμενή νημάτων (thread pool) και μια συνεχόμενη ροή από ανεξάρτητες εργασίες (jobs). Όταν δημιουργείται μια εργασία, μπαίνει στην ουρά προτεραιότητας του χρονοπρογραμματιστή και περιμένει να εκτελεστεί. Οι εργασίες εκτελούνται με την σειρά που δημιουργήθηκαν (first-in-first-out - FIFO). Κάθε νήμα περιμένει στην ουρά μέχρι να του ανατεθεί μια εργασία και, αφού την εκτελέσει, επιστρέφει στην ουρά για να αναλάβει νέα εργασία. Για την ορθή λειτουργία ενός χρονοπρογραμματιστή είναι απαραίτητη η χρήση σημαφόρων στην ουρά, ώστε να μπλοκάρουν εκεί τα νήματα, και την κρίσιμη περιοχή (critical section), ώστε να γίνεται σωστά εισαγωγή και εξαγωγή εργασιών από την ουρά.
- Η υλοποίηση του χρονοπρογραμματιστή έχει γίνει με την δομή Queue
- Τις μεθόδους του χρονοπρογραμματιστή που επιτυγχάνουν το ασφαλή συγχρονισμό των νημάτων.

Έχουμε δημιουργήσει δύο χρονοπρογραμματιστές, έναν που παραλληλοποιεί τον υπολογισμό των gradients στα batches (θα εξηγηθεί σε επόμενο κεφάλαιο η λειτουργία αυτή) και έναν που δημιουργεί παράλληλα ζευγάρια για να γίνει το retrain.
Οφέλη αυτής της επιλογής:

1. Δεν γίνεται κατάχρηση του αριθμού των threads (καθώς επαναχρησιμοποίουνται) και εκμεταλευόμαστε την όλη επιτάχυνση που μπορούν να προσφέρουν.
2. Εργασίες ανεξάρτητες όπως αυτές που προανέφερα παραλληλοποιούνται με αποτέλεσμα την επιτάχυνση.

4.5 Σύστημα testing

Για τον έλεγχο των δομών αλλά και των μοντέλων που έχουμε δημιουργήσει χρησιμοποιούμε την βιβλιοθήκη acutest.h
Στον φάκελο tests υπάρχουν test μέθοδοι για τα κυριότερα κομμάτια του κώδικα που θα μπορούν να ελεγχθούν με αυτόν τον τρόπο.

Περισσότερες πληροφορίες για την βιβλιοθήκη acutest Link

5 Μοντέλο μηχανικής μάθησης - Logistic Regression

5.1 Αλγόριθμος Logistic Regression

Η λογιστική παλινδρόμηση είναι ένας αλγόριθμος ταξινόμησης μηχανικής μάθησης που χρησιμοποιείται για να εκτιμήσει σε ποια από τις ορισμένες τάξεις ανήκει μία παρατήρηση/δείγμα. Παραδείγματα χρήσης είναι να ο διαχωρισμός email σε spam/ham, διάκριση καλοήθειας/κακοήθειας σε ακτινογραφίες, κ.ο.κ. Η λογιστική παλινδρόμηση που θα χρησιμοποιήσουμε θα είναι δυαδική.

Το μοντέλο λογιστικής παλινδρόμηση είναι παρόμοιο με το μοντέλο γραμμικής παλινδρόμησης, αλλά με μία πολυπλοκότερη συνάρτηση σφάλματος. Η συνάρτηση σφάλματος είναι η λογιστική συνάρτηση (logistic or sigmoid function) αντί για μία γραμμική συνάρτηση. Προκειμένου να αντιστοιχίσουμε τις τιμές πρόβλεψης σε πιθανότητες, χρησιμοποιούμε τη λογιστική συνάρτηση. Η συνάρτηση αντιστοιχίζει κάθε τιμή πραγματικού αριθμού σε μία άλλη τιμή μεταξύ 0 και 1.

Θα υλοποιήσουμε τη λογιστική παλινδρόμηση μιας εξαρτημένης μεταβλητής στο σύνολο των εξαρτημένων μεταβλητών $x = (x_1, x_2, x_3, ...)$, όπου είναι το πλήθος των διαστάσεων του διανύσματος εισόδου. Ξεκινάμε με τις γνωστές τιμές των διανυσμάτων εισόδου και την αντίστοιχη πραγματική απόκριση , για κάθε γνωστή παρατήρηση i = 0,1,2,...,n .

Στόχος είναι να βρεθεί η συνάρτηση λογιστικής παλινδρόμησης $p(x)$, ώστε οι αποκρίσεις της πρόγνωσης $p(x_i)$ να είναι όσο εγγύτερα γίνεται στην πραγματική απόκριση y_i για κάθε παρατήρηση i = 0,1,2,...,n . Υπενθυμίζεται ότι η πραγματική απόκριση μπορεί να είναι μόνο 0 ή 1 σε προβλήματα δυαδικής κατηγοριοποίησης. Αυτό σημαίνει ότι κάθε $p(x_i)$ θα πρέπει να είναι κοντά είτε στο 0 είτε στο 1. Αυτός είναι και ο λόγος που χρησιμοποιείται η σιγμοειδής συνάρτηση.

Από τη στιγμή που θα οριστικοποιηθεί η συνάρτηση λογιστικής παλινδρόμησης $p(x)$, μπορούμε να τη χρησιμοποιήσουμε για την πρόγνωση εξόδων για νέες και άγνωστες εισόδους, θεωρώντας ότι οι υποκείμενες μαθηματικές εξαρτήσεις δεν έχουν αλλάξει.

Η υλοποίηση του μοντέλου, γίνεται με βάση τους παρακάτω θερητικούς τύπους

- Παραγωγισμένη συνάρτηση σφάλματος

$$\nabla J(w, b) = \sum_{i=1}^{n} (\sigma(w^T x^{(i)} + b) - y^i) x_j^i)$$

- Ενημέρωση βαρών

$$w^{t+1} = w^t - \eta * \nabla J(w, b) \text{ ,όπου } \eta\text{: } learning\ rate$$

5.2 Μελέτη σύγκλισης του αλγορίθμου

Η σύγκλιση του αλγορίθμου παρατηρείται και ελέγχεται με δύο τρόπους:

- **Αριθμός μέγιστων εποχών**, ο οποίος φροντίζει το μοντέλο να μην υπερβεί αυτές τις εποχές σε περίπτωση που δεν υπάρχει σύγκλιση.
- **Ευκλείδια νόρμα**,την υπολογίζουμε για το διάνυσμα των βαρών σε κάθε εποχή έτσι ώστε να φτάσουμε σε σύγκλιση. Ταυτόχρονα, δίνουμε και ένα κατώφλι το οποίο μας υποδηλώνει τη σύγκλιση σε περίπτωση που η νόρμα των αρχικών βαρών αφαιρούμενη από την νόρμα των βαρών ύστερα απο μια εποχή, δεν ξεπέρνα το κατώφλι. Παραθέτουμε και το τύπο της ευκλείδιας νόρμας:

$$||x||_2 = \sqrt{x_1^2 + x_2^2 + ... + x_n^2}$$

5.3 Gradient Descent

5.3.1 Full-Batch GD

Σε αυτήν την υλοποίηση κάθε υπολογίσμος των μερικών παραγώγων και γενικότερα του gradient χρειαζόταν προσπέλαση ολόκληρου του Dataset, γεγονός το οποίο παρατηρήσαμε ότι:

- Βελτίωνε και επιτάχυνε σε εποχές (δηλαδή λιγότερες εποχές για σύγκλιση) την διαδικασία σύγκλισης, παρουσιάζοντας ομαλότερες μεταβάσεις, ώστοσο αποδείχθηκε κακή τακτική λόγω του πολύ με μεγάλου όγκου επαναλήψεων σε κάθε εποχή.
- Επέφερε τεράστια αύξηση στον χρόνο εκπαίδευσης του μοντέλου.

Η υλοποίηση Full-Batch Gradient Descent, ήταν η πρώτη μας προσπάθεια, την οποία απορρίψαμε καθώς δεν ήταν ζητούμενη, αλλά και γιατί, με βάση τους πειραματισμούς μας, χρειαζόταν υπερβολικά μεγάλο χρόνο για να εκτελεστεί.

5.3.2 Stochastic GD

Ο αλγόριθμος αυτός υλοποιήθηκε ώς μέρος του δεύτερου σκέλους υλοποίησης και συνοπτικά, το μοντέλο δέχεται τα διανυσματοποίημενα δεδομένα και για καθένα απο αυτά βρίσκει το loss και το gradient και ενημερώνει τα αποθηκεύμενα βάρη. Για αυτόν τον αλγόριθμο παρατηρούμε τα εξής:

- Χρείαζεται περισσότερες εποχές σε σχέση με του άλλους για να επέλθει σύγκλιση, αλλά και η προσπάθεια σύγκλισης δεν είναι ομαλή λόγω της συνεχόμενης αλλαγής των βαρών.
- Μείωσε σημαντικά τον χρόνο εκπαίδευσης του μοντέλου.

5.3.3 Mini-Batch GD

Mini-Batch GD είναι ο τελικός μας αλγόριθμος, ο οποίος έχει υλοποίηθει και υπάρχει στην τελική υλοποίηση του συστήματος μας. Συνοπτικά, αυτός ο αλγόριθμος είναι ένα υβρίδιο των δύο προηγούμενων καθώς ενημερώνει τα βάρη του αλγορίθμου για κάθε δέσμη δεδομένων. Επιλέγεται ώς μέγεθος δέσμης τα 516,10264 και 2056 στοιχεία, οπού με βάση αυτά έγιναν και οι πειραματισμοί μας. Αυτός ο αλγόριθμος μας προσφέρει τα καλύτερα αποτελέσματα και για τον οποίον κανούμε τις εξής παρατηρήσεις:

- Παρουσιάζει μια πιο ομαλή συμπεριφορά σύγκλισης απο τον stochastic αλγόριθμο.
- Ο χρόνος εκτέλεσης δεν είναι μεγάλος.

5.4 Υλοποίηση

Η υλοποίηση των ανωτέρω υπάρχει στον φάκελο /modules/LogisticRegression και αποτελείται απο τις εξής μεθόδους,

```
1 LogisticRegression* LR_construct(..);
2 void LR_fit(..);
3 int LR_predict(..);
4 void LR_Evaluation(..);
5 float LR_predict_proba(.);
6 int decision_boundary(.);
7 float sigmoid(..);
8 float euclid_norm(..);
```

Γράφημα 5: Κύριες λειτουργίες μοντέλου

Συνοπτικά η λειτουργία του μοντέλου:

1. Δέχεται ένα σύνολο απο concated διανύσματα σε sparse μορφή.
2. Για κάθε δέσμη δεδομένων αναθέτει στον JobScheduler (threads) την εύρεση του διανύσματος μερικών παραφώγων (gradients)
3. Όταν τελείωσουν ένας αριθμός απο batches-threads συγκεντρώνεται από όλα αυτά τα διανύσματα και υπολογίζεται ένας μέσος όρος για κάθε βάρος του μοντέλου.
4. Με αυτούς τους μέσους όρους πολλαπλασιασμένους με το learning rate ανανεώνονται τα βάρη και το bias και επαναλαμβάνεται αυτή η διαδικασία μεχρί εξαντλήσεως

των batches (δηλαδή όλων των δεδομένων).

5.5 Εκτίμηση απόδοσης μοντέλου

Το πρόβλημα της κατηγοριοποίησης που έχουμε να αντιμετωπίσουμε είναι δυαδικό. Αυτό σημαίνει πως μας ενδιαφέρουν οι εξής ποσότητες:

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Γράφημα 6: Επεξήγηση όρων

Η εκτίμηση της απόδοσης του μοντέλου γίνεται μέσω των εξής μετρικών:

- **Accuracy:** Αυτή η μετρική, μας δείχνει πόσα σώστες εκτιμήσεις, θετικές και αρνητικές έχει κάνει το μοντέλο. Ο τύπος υπολογισμού της είναι:

$$\textbf{Accuracy} = \frac{\textbf{True\_Positive} + \textbf{True\_Negative}}{\textbf{True\_Positive}+\textbf{True\_Positive}+\textbf{False\_Negative}+\textbf{False\_Positive}}$$

- **Precision:**

$$\textbf{Precision} = \frac{\textbf{True\_Positive}}{\textbf{True\_Positive} + \textbf{False\_Positive}}$$

- **Recall:**

$$\textbf{Recall} = \frac{\textbf{True\_Positive}}{\textbf{True\_Positive} + \textbf{False\_Negative}}$$

- **F1-Score:**

$$\textbf{F1-Score} = 2 * \frac{\textbf{Precision*Recall}}{\textbf{Precision} + \textbf{Recall}}$$

Οι μετρικές αυτές μας δείχνουν την απόδοση του μοντέλου όχι μόνο για την μια κατηγορία προβλέψεων αλλά και για τις δύο, δείνοντάς μας μια πολύ καλύτερη είκονα της κατηγοριοποίησης που γίνεται.

5.6 Επανεκπαίδευση μοντέλου

Η επανεκπαίδευση του μοντέλου, επιτυγχάνεται μέσω ενός while loop που διαρκεί είτε ορισμένες επαναλήψεις, είτε σταματά για συγκεκριμένες τιμές του threshold. Αρχικά, εκπαιδεύεται το μοντέλο μέσω του αλγορίθμου Mini batch GD του logistic regression με χρήση πολυνηματισμού.

Στη συνέχεια, πάλι με χρήση πολυνηματισμού, διατρέχονται όλα τα ζευγάρια που δεν είχε καθοριστεί θετική ή αρνητική συσχέτιση και δίνονται στο μοντέλο. Αν προκύψει ισχυρή θετική ή αρνητική πιθανότητα, το νέο ζευγάρι αποθηκεύεται σε έναν πίνακα.

Έπειτα ο πίνακας αυτός ταξινομείται με την qsort από την μεγαλύτερη πιθανότητα στην μικρότερη. Μέσω της συνάρτησης resolve-transitivity-issues ορισμένα ζεγγάρια ενσωματώνονται στο training set.

5.7 Πολυνηματισμός

Για τον γρηγορότερο υπολογισμό των βαρών, χρησιμοποιείται η επιτάχυνση-παραλληλοποίηση μέσω threads (pthreads). Για την αναγκή αυτήν έχουμε δημιουργήσει τον JobScheduler στον οποίο ανατίθονται batches για τα οποία πρέπει να υπολογίσει το διανύσμα των Gradients. Κάθε φορά που ένας αριθμός νημάτων έχει ολοκλήρωσει αυτόν τον υπολογιμό, βρίσκουμε τους M.O και ανανεώνουμε τα βάρη του μοντέλου. Η επεξήγηση της δουλείας του JobScheduler παραλείπεται σε αυτό το σημείο, καθώς έχει αναλύθει σε προηγούμενη κατηγόρια.

5.8 Καλύτεροι παράμετροι

5.8.1 Χωρίς Retrain

Στον παρακάτω πίνακα βλέπουμε τις 5 περιπτώσεις με το μεγαλύτερο accuracy score, στην περίπτωση που το μοντέλο μας δεν έχει γίνει retrain, με δεδομένο το medium csv αρχείο. Ο αριθμός των thread ήταν σταθερός σε όλες τις περιπτώσεις, ενώ για τις οπόλοιπες μεταβλητές δοκιμάστηκαν διαφορετικές τιμές.

Learning rate	threshold euclid	Epochs	batch size	Threads	Test Accuracy	Test Recall	Test Precision	Test F1	Valid Accuracy	Valid Recall	Valid Precision	Valid F1	Time CPU	Time Real
0.001	0.00010	50	2056	10	92.69	8.50	70.93	15.17	92.74	8.36	75.00	15.04	1.41	1.52
0.001	0.00001	50	2056	10	92.69	8.50	70.93	15.17	92.74	8.36	75.00	15.04	1.41	1.54
0.001	0.00100	50	2056	10	92.69	8.50	70.93	15.17	92.74	8.36	75.00	15.04	1.39	1.53
0.100	0.10000	50	2056	10	92.45	16.02	53.00	24.60	92.68	17.41	58.14	26.80	1.40	1.50
0.100	0.00100	50	2056	10	92.45	16.02	53.00	24.60	92.68	17.41	58.14	26.80	1.35	1.44

Πίνακας 1: Καλύτερα αποτελέσματα για το medium

Από τα παραπάνω δεδομένα, παρατηρούμε πως τα υψηλότερα accuracy score επιτυγχάνονται με μεγάλο πλήθος εποχών και με μεγάλο batch size. Παράλληλα, στις τρεις πρώτες περιπτώσεις το learning rate είναι το μικρότερο από αυτά που δοκιμάστηκαν.

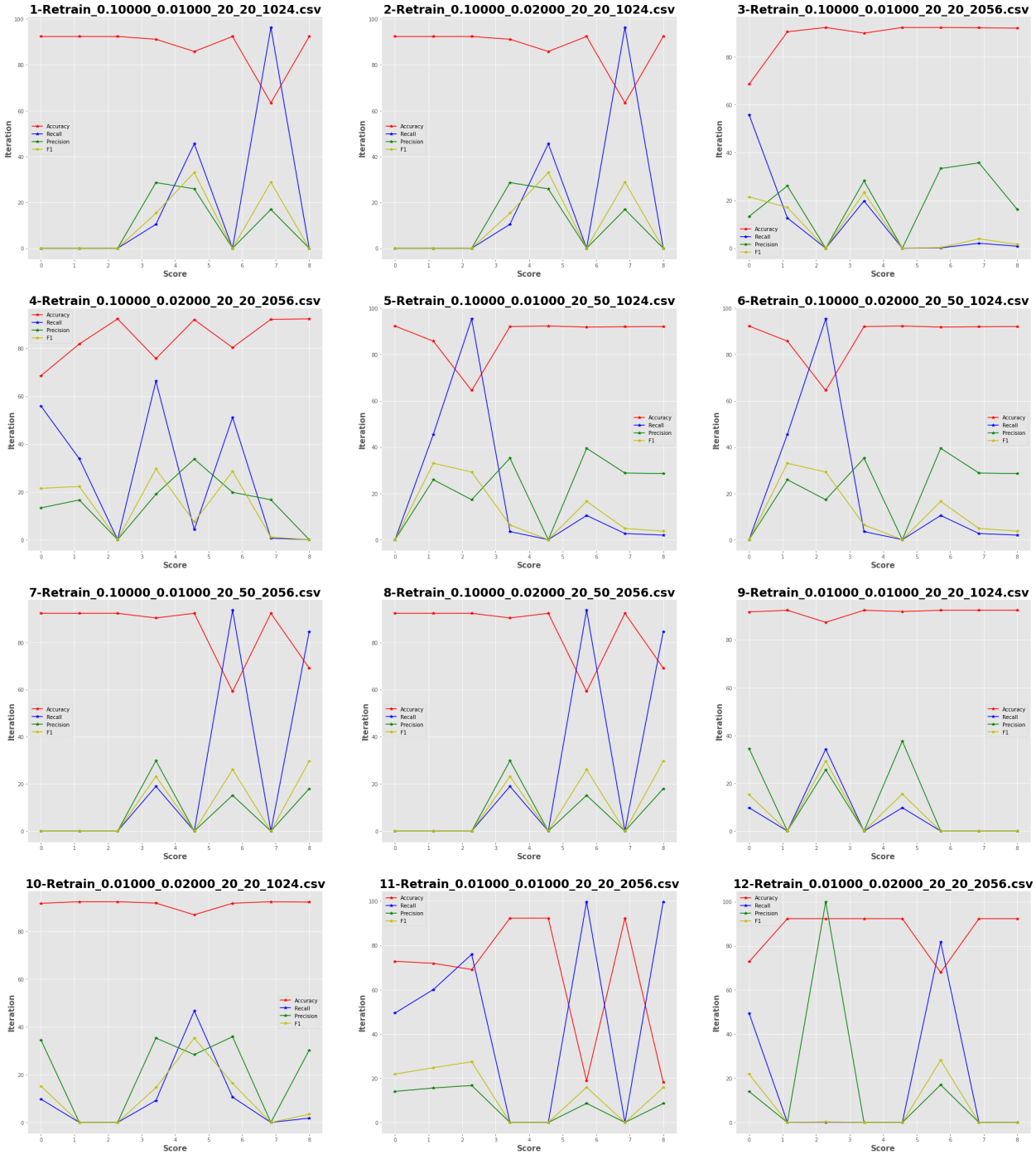
Στον παρακάτω πίνακα βλέπουμε τις 5 περιπτώσεις με το μεγαλύτερο accuracy score, στην περίπτωση που το μοντέλο μας δεν έχει γίνει retrain, με δεδομένο το large csv αρχείο.

Learning rate	threshold euclid	Epochs	batch size	Threads	Test Accuracy	Test Recall	Test Precision	Test F1	Valid Accuracy	Valid Recall	Valid Precision	Valid F1	Time CPU	Time Real
0.001	0.00001	50	512	20	88.52	12.06	75.64	20.81	88.61	12.60	77.26	21.66	9.85	10.48
0.001	0.00010	50	512	20	88.52	12.06	75.64	20.81	88.61	12.60	77.26	21.66	9.86	13.65
0.001	0.00100	50	512	20	88.52	12.06	75.64	20.81	88.61	12.60	77.26	21.66	9.86	10.50
0.001	0.10000	50	512	20	88.49	11.50	76.44	19.99	88.56	11.97	77.68	20.74	9.24	9.85
0.010	0.00001	50	512	10	88.42	9.79	80.02	17.45	88.44	9.99	80.34	17.77	10.00	10.62

Πίνακας 2: Καλύτερα αποτελέσματα για το large

Όπως και στον προηγούμενο πίνακα, παρατηρούμε πως τα υψηλότερα score εντοπίζονται σε εκτελέσεις με 50 εποχές και χαμηλό learning rate. Από το γεγονός αυτό, μπορούμε να υποθέσουμε πως το χαμηλό learning rate βοηθάει στην εκπαίδευση του μοντέλου, ενδεχομένως, λόγω του μεγάλου πλήθους δεδομένων.

5.8.2 Με Retrain



Γράφημα 7: Γραφήματα μετρικών Accuracy,Recall,Precision και F1 σε κάθε επανάληψη του retrain εφαρμοσμένα στο test set και με βάση το Dataset Y.O τίτλος του κάθε γραφήματος αντιστοιχεί στο αρχείο με τα αποτελέσματα που ανήκει στον φάκελο Retrain_Data.

Learning rate	Threshold euclid	Threshold retrain	Epochs	Batch size	Threads	Valid Accuracy	Valid Recall	Valid Precision	Valid F1	Time CPU	Time Real
0.00100	0.0001	0.020000	50	2056	20	92.80	14.62	64.02	23.81	5.61	10.20
0.001000	0.000100	0.020000	50	1024	20	92.37	3.34	57.14	6.32	13.45	18.61
0.001000	0.000100	0.010000	50	1024	20	92.37	3.34	57.14	6.32	13.57	18.81
0.10000	0.000	0.020	5	1024	20	92.31	0.00	0.00	0.00	1.85	5.33

Πίνακας 3: Καλύτερα αποτελέσματα με χρήση της μεθόδου retrain

Συνοπτικά η λειτουργία του retrain:

1. Διατρέχεται ένα μέρος των πιθανών ζευγαριών που προκύπτουν απο τις κάμερες (με χρήση threads)
2. Τα ζευγάρια που το μοντέλο προβλέπει με ισχυρή πιθανότητα, δηλαδή με πιθανότητα μεγαλύτερη απο 1-threshold ή μικρότερη απο threshold (όπου threshold ένας αριθμός στο διάστημα [0.0,0.1]), εισάγονται στα δεδομένα εκπαίδευσης και το μοντέλο επανεκπαιδεύται με αυτά σε επόμενη επανάληψη.
3. Επιλύονται διάφορες αντικρουόμενες εισαγωγές
4. Επαναλαμβάνεται αυτή η διαδικασία (κάθε φόρα και για διαφορετικούς συνδιασμούς ζευγαριών)

* Τα ζευγάρια για τα οποία γίνεται retrain είναι ανεξάρτητα απο αυτά του validation ή του test καθώς έτσι τα αποτελέσματα του retrain είναι πιο ρεαλιστικά.

Γενικές παρατηρήσεις για το re-train:

- Το μοντέλο που έχουμε υλοποίησει έχει εκπαιδευτεί με σημαντικά μικρότερο αριθμό ίδιων ζευγαριών απο ότι διαφορετικών, κάτι που ευθύνεται για την "κακή" απόδοση στις μετρικές Precision,Recall και F1. Με το retrain θέλουμε να είσαχθούν ίδια κυρίως ζευγάρια για να αποκτήσει το μοντέλο μεγαλύτερη ακρίβεια στην πρόβλεψη των ίδιων ζευγαριών.
- Με βάση τους πειραματισμούς μας, συμπεράναμε ότι η μέθοδος retrain δεν είχε τα επιθυμητά αποτελέσματα, εκτός ελάχιστων περιπτώσεων.

Παρατηρήσεις με βάση τα διαγράμματα:

- Η κόκκινη καμπύλη (Accuracy) ότι κυρίως παραμένει στάσιμη, κάτι που σημαίνει ότι το retrain τελικά δεν αυξάνει σημαντικά την ακρίβεια. (πχ Διάγραμμα 3 στο Γράφημα 7)
- Σε αλλές περιπτώσεις όμως, λόγω λανθασμένων προβλέψεων που άλλαξαν την μορφή των κλικών, η κόκκινη καμπύλη (Accuracy) παρουσιάζει μεγάλες πτώσεις και ανόδους,με αποτέλεσμα το μοντέλο να γίνεται χειρότερο απο ότι ήταν στην αρχή. (πχ Διάγραμμα 11,8 στο Γράφημα 7)

- Η καμπύλη Recall (μπλέ) έχει αντίστορφη συμπριφορά απο ότι η καμπύλη Accuracy (κόκκινη), κάτι αναμενόμενο καθώς όταν το μοντέλο προβλέπει καλύτερα τα ίδια ζευγάρια, χάνει ακρίβεια απο τα διαφορετικά αντικείμενα.(πχ Διάγραμμα 3,5,6 στο Γράφημα 7)
- Οι καμπύλες Recall,Precision,F1 παραμένουν χαμηλότερα της κόκκινης καθώς υπάρχει αδυναμία στην πρόβλεψη των ίδιων ζευγαριών.

Παρατηρήσεις με βάση τον πίνακα αποτελεσμάτων:

- Μόνο σε μια περίπτωση το μοντέλο προβλέπει όλα τα μηδενικά και κάποιους άσσους (ακρίβεια 92.80%)
- Το batch με μέγεθος 2056 προσφέρει και πάλι τα καλύτερα αποτελέσματα όπως και χωρίς retrain.

** Όλα τα αποτελέσματα αυτά έχουν γίνει με χρήση του αρχείου-set medium.

6 Εναλλακτική υλοποίηση δεύτερου σκέλους

Στο δεύτερο σκέλος υλοποίησης του συστήματος κληθήκαμε να υλοποίησουμε την τακτική κατηγοριοποίησης με έναν εναλλακτικό τρόπο. Πιο συγκεκριμένα:

- Κάθε κλίκα εκπαίδευεται ξεχωρίστα με τα δικά της αντικείμενα και συνεπώς δημιουργείται ένα μοντέλο κατηγοριοποίησης για κάθε κλίκα.
- Ύστερα σχηματίζεται ένα σύνολο απο αντικείμενα απο όλες τις κλίκες και δοκιμάζεται κάθε αντικείμενο σε κάθε κλίκα. Η κλίκα που θα δώσει την μεγαλύτερη πιθανότητα, θα είναι και αυτή που ανήκει το αντικείμενο.

Η συγκεκριμένη υλοποίηση, απορρίφθηκε λόγω της πολύ μικρής απόδοσης που είχε, αλλά και γιατί τελικά δεν ήταν ζητούμενο.

7 Βελτιώσεις χρήσης μνήμης και χρόνου εκτέλεσης

Στα διάφορα στάδια της εργασίας παρατηρήθηκε πως οι εξής σχεδιαδτικές επιλογές βελτίωσαν τον χρόνο εκτέλεσης του προγράμματος, χωρίς επιβάρυνση της μνήμης:

- **Bit-array**: Για να αποφευχθεί η αποθήκευση περιττών πληροφοριών αλλά και η συνεχής προσπέλαση της μνήμης, χρησιμοποιήθηκε bit Array όπου ήταν δυνατό, π.χ. αποθήκευση θετικών και αρνητικών συσχετίσεων.
- **Disjoint set**: Για τον σχηματισμό των κλικών χρησιμοποιήθηκε disjoint set, έναντι της χρήσης λιστών και δεικτών. Ο χώρος που δεσμεύτηκε ήταν μικρότερος και οι υπολογισμοί αισθητά ταχύτεροι.
- **Dictionary**: Με σκοπό την αποφυγή πολλαπλών προσπελάσεων των json, όλες οι απαραίτητες πληροφορίες συλλέγονται και όλοι οι δυνατοί υπολογισμοί γίνονται στο πρώτο διάβασμα των αρχείων.
- **Sparse Matrix**: Για την βέλτιστη αποθήκευση των διανυσμάτων γίνεται η αποθήκευση μόνο των μη αρνητικών θέσεων του διανύσματος. Με αυτόν τον τρόπο το διάβασμα, αλλά και η αποθήκευσή τους γίνονται ταχύτερα.
- **Red Black Tree**: Στις περιπτώσεις που χρειάστηκε να γίνει αναζήτηση δεδομένων, χρησιμοποιήθηκε red black tree, αφού για την αναζήτηση, στην χειρότερη περίπτωση έχει πολυπλοκότητα O(logn), ενώ αντίθετα άλλες δομές έχουν O(n).

8 Αναφορές

[1.] Hackerearth ορισμός disjoint set

[2.] Διαφάνειες Threads, K24:ΣΥΣΠΡΟ

[3.] Αλγόριθμος RBT, Introduction to Algorithms Third Edition, T. Cormen, C. Leiserson, R. Rivest, C. Stein