Κ23Γ : Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα

Χειμερινο Εξαμηνο 2020-21

3η Προγραμματιστική Εργασία

Βασιλοπουλου Θωμαϊς (1115201500016)
Γεωργακλη Έλενα (1115201500023)

Ερωτημα Α:

Α1. Δομη Παραδοτεου

Το παραδοτέο για το ερώτημα 3Α αποτελείται από τα ακόλουθα αρχεία:

- reduce.py: το εκτελέσιμο που υλοποιεί το ζητούμενο Α
- results: φάκελος που περιέχει τα γραφικά αποτελέσματα της εκτέλεσης του προηγούμενου αρχείου, καθώς και αρχεία με τα αποτελέσματα των πειραμάτων σε κάθε περίπτωση

Α2. Λειτουργικοτητα Εργασιας

Το εκτελέσιμο τρέχει με την εντολή:

```
python reduce.py -d train-images.idx3-ubyte -q t10k-images.idx3-ubyte -od
outputDataset.txt -oq outputQueryset.txt
```

Με την εκκίνηση της εκτέλεσης γίνεται έλεγχος για την ορθότητα των ορισμάτων και στην συνέχεια ζητούνται από τον χρήστη οι υπερπαράμετροι για την παρούσα υλοποίηση του autoencoder. Όταν αντληθούν όλες οι απαραίτητες τιμές από το χρήστη, ο έλεγχος μεταβαίνει στην συνάρτηση:

```
def autoencoderBottleneck(dataset, queryset, layers, maxFilters, x, y, convFiltSize, ba
tchSize, epochs, latentDim)
```

Πιο αναλυτικά, αντλούνται τα metadata από το αρχείο εισόδου (magic number, number of images, dx, dy), διαβάζονται οι εικόνες, μετατρέπει αυτά που διαβάστηκαν σε numpy array των κατάλληλων διαστάσεων και ορίζει το σχήμα του μοντέλου. Στην συνέχεια δημιουργείται το ζητούμενο μοντέλο με σκοπό να αξιολογηθεί για την απόδοσή του, με μετρική απώλειας mean_squared_error. Για την ολοκλήρωση της προηγούμενης διαδικασίας λαμβάνει χώρα η διαδοχική κλήση του encoder και decoder, για τους οποίους ισχύουν τα ακόλουθα:

a. `encoder(inputImg, layers, maxFilters = 64, convFiltSize = (3, 3)) ))`

Αποτελείται από layers, που δύναται να είναι είτε Conv2D, είτε MaxPooling2D ή BatchNormalization. Σκοπός του encoder είναι η εικόνα που θα χρησιμοποιηθεί στην συνέχεια από τον decoder να έχει ίδιες διαστάσεις με την αρχική. Για τον λόγο αυτό, το MaxPooling2D λαμβάνει χώρα μετά τα δύο πρώτα BatchNormalization και όχι αργότερα.

b. `decoder(conv, layers, maxFilters, convFiltSize, dx, dy)`

Αποτελείται από layers, που δύναται να είναι είτε Conv2DTranspose, είτε UpSampling2D ή BatchNormalization. Ο decoder είναι απόλυτα συμμετρικός με τον encoder, το UpSampling2D λαμβάνει χώρα μετά τα δύο τελευταία BatchNormalization και όχι νωρίτερα.

Αξίζει να αναφερθεί ότι τα μοντέλα που παράγονται τόσο από τον encoder, όσο και από τον decoder δεν παρουσιάζουν overfit.

Όταν ο encoder και ο decoder ολοκληρώσουν την εκτέλεσή τους, δημιουργείται input κατάλληλων διαστάσεων και διαχωρίζεται το training set, ώστε το 80% να είναι για train και το 20% να είναι για validation. Στο μοντέλο του autoencoder ως ground truth ορίζεται το ίδιο το σύνολο των εικόνων και κανονικοποιούνται οι τιμές ώστε να είναι στο διάστημα [0.0, 1.0]. Στην συνέχεια εκπαιδεύεται το παραχθέν μοντέλο με βάσει τις δοθείσες από τον χρήστη υπερπαραμέτρους.

Μόλις έχει κατασκευαστεί το επιθυμητό μοντέλο, ξεκινάει η διαδικασία ανάγνωσης του query set από το δοθέν αρχείο. Αρχικά, αντλούνται τα metadata (magic number, number of images, dx, dy), διαβάζονται οι εικόνες και το αποτέλεσμα μετατρέπεται σε numpy array των κατάλληλων διαστάσεων. Τέλος διαχωρίζεται το query set, ώστε το 80% να είναι για train και το 20% να είναι για validation και εκπαιδεύεται το παραχθέν μοντέλο με τις προηγούμενες υπερπαραμέτρους.

Με την ολοκλήρωση του autoencoder, παράγονται οι εικόνες τόσο για το data set, όσο και για το query set στον νέο διανυσματικό χώρο (10x10) από την συνάρτηση:

```
def reduce(outputDataset, outputQueryset, latentDim, xTrain, xQuery)
```

A3. Αποτελεσματα Εκτελεσεων

Στην συνέχεια παρατίθενται μετρήσεις κατά τις οποίες, για τον εντοπισμό αλλαγών καθώς διαφοροποιούνται οι υπρεπαράμετροι. Οι προκαθορισμένες τιμές που χρησιμοποιούνται στα πειράματα για τις μεταβλητές είναι οι ακόλουθες:

- *layers = 20*
- *convFiltSize = (3,3)*
- *maxFilterSize = 64*
- *epochs = 50*
- *batchSize = 128*
- *latentDim = 10*

Σε κάθε μία από τις γραφικές παραστάσεις που ακολουθούν γίνεται σύγκριση του σφάλματος που προκύπτει κατά την εκτέλεση του autoencoder με training και validation set. Γενικά είναι σημαντικό να αναφερθεί, ότι το σφάλμα κινούνταν γύρω από το ίδιο εύρος τιμών σε κάθε περίπτωση πειραματισμού για το training set και οι διαταραχές ήταν ορισμένες φορές ευκολότερα ορατές στο σφάλμα του validation set.

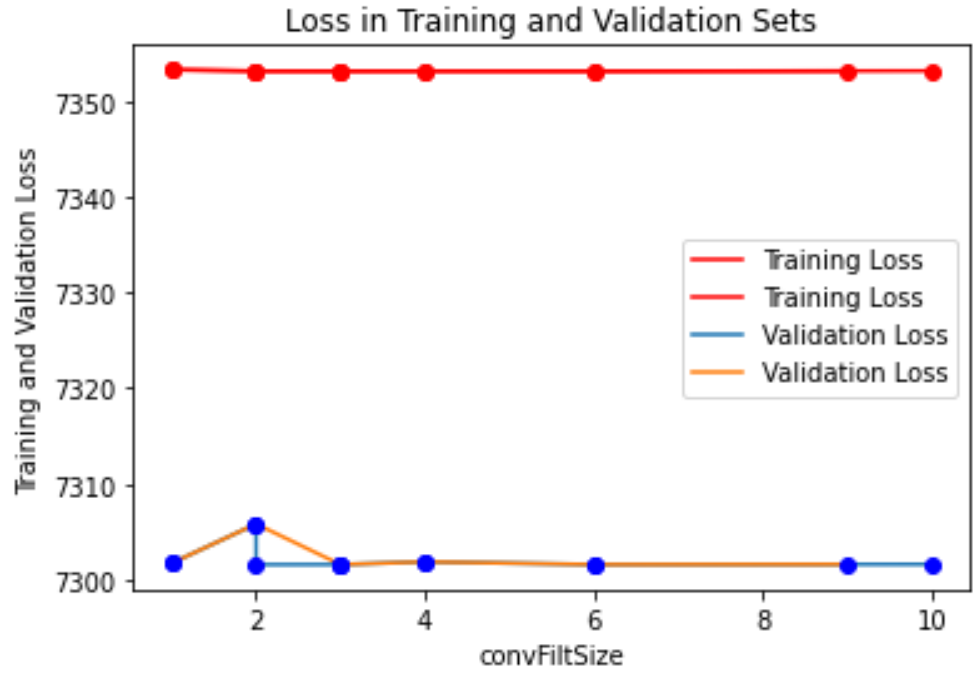
Είναι σημαντικό να αναφερθεί, ότι το πλήθος των συνελικτικών στρωμάτων μοιράζεται ισάξια στον encoder και τον decoder. Για παράδειγμα, αν δοθεί πλήθος layers = 12, τότε ο encoder θα χρησιμοποιήσει τα 6 από αυτά και ο decoder τα υπόλοιπα 6. Αξίζει, επίσης, να αναφερθεί, ότι το ελάχιστο πλήθος των συνελικτικών στρωμάτων, για να μην προκύψει υπερχείλιση, είναι 12, δεδομένου, ότι ο encoder και ο decoder χρησιμοποιούν κατ’ ελάχιστο 6 ο κάθε ένας.

No 1: Πληθος Συνελικτικων Στρωματων

Εικόνα 1: Διαμόρφωση Σφάλματος για διάφορα πλήθη συνελικτικών στρωμάτων

Παρατηρείται, ότι το πλήθος των συνελικτικών στρωμάτων δεν επηρεάζει σημαντικά την απόδοση του autoencoder, όσον αφορά το σφάλμα. Παρόλα αυτά, όσο αυξάνεται το πλήθος των layer παρατηρείται και μία μικρή αύξηση στο αντίστοιχο σφάλμα και στις δύο περιπτώσεις.

No 2: Μεγεθος Συνελικτικων Φιλτρων



Εικόνα 2: Διαμόρφωση Σφάλματος για διάφορα πλήθη συνελικτικών φίλτρων

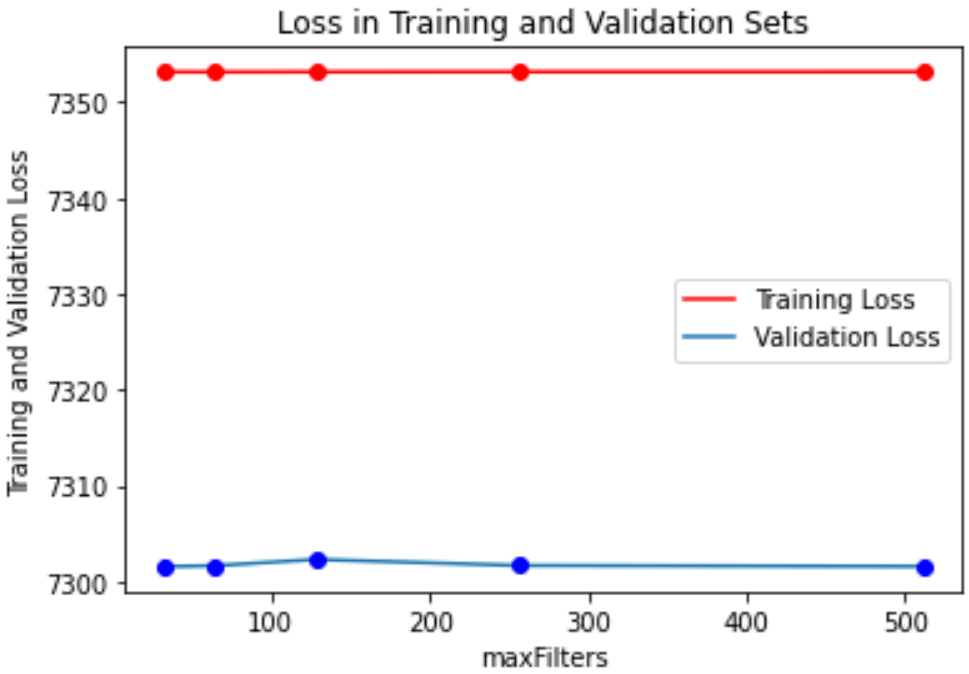
Από το παραπάνω διάγραμμα παρατηρείται, ότι η απόδοση του autoencoder δεν επηρεάζεται σημαντικά από το πλήθος των συνελικτικών φίλτρων. Όμως, για φίλτρο διάστασης (2,2) υπάρχει μία αύξηση στο

σφάλμα του validation set. Τέλος, αξίζει να αναφερθεί, ότι όσο αυξάνεται η διάσταση του συνελικτικού φίλτρου, τόσο αυξάνεται και ο χρόνος εκτέλεσης του προγράμματος.

Οι διαστάσεις των φίλτρων που χρησιμοποιήθηκαν για την πειραματική μελέτη είναι οι ακόλουθες:

- (1,1)
- (2,2)
- (2,3)
- (3,3)
- (4,4)
- (6,6)
- (10,9)

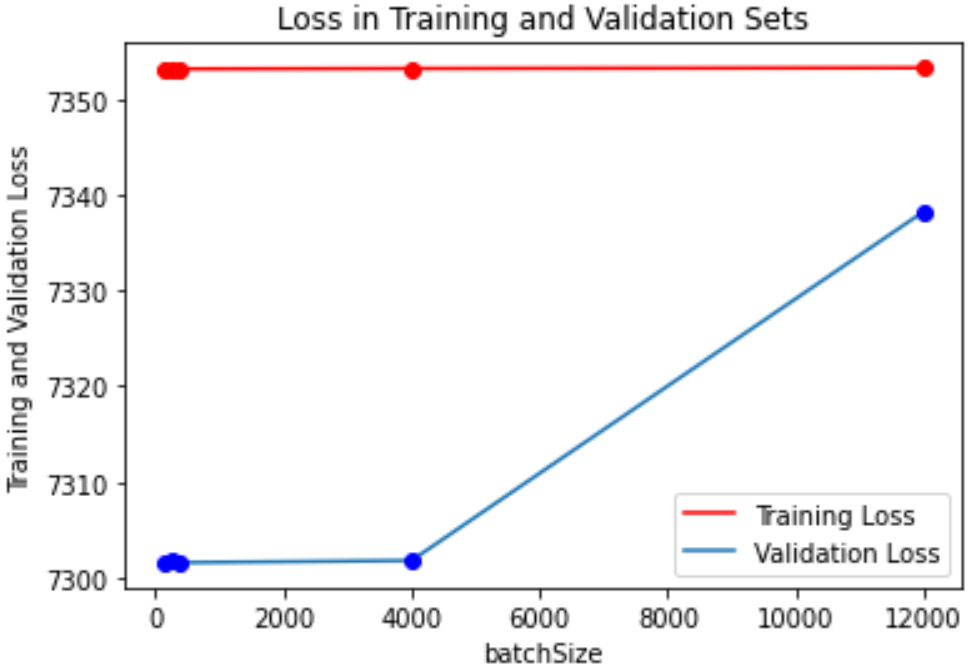
No 3: Μεγιστο Πληθος Φιλτρων ανα Στρωμα



Εικόνα 3: Διαμόρφωση Σφάλματος για διάφορες τιμές μέγιστου πλήθους συνελικτικών φίλτρων

Όπως φαίνεται από το παραπάνω διαγράμμα, η αύξηση του μέγιστου πλήθους των συνελικτικών φίλτρων, δεν επηρεάζει την απόδοση του autoencoder ως προς το σφάλμα. Παρόλα αυτά, όσο αυξάνεται το μέγεθος δέσμης, τόσο επιταχύνεται η εκτέλεση του προγράμματος.

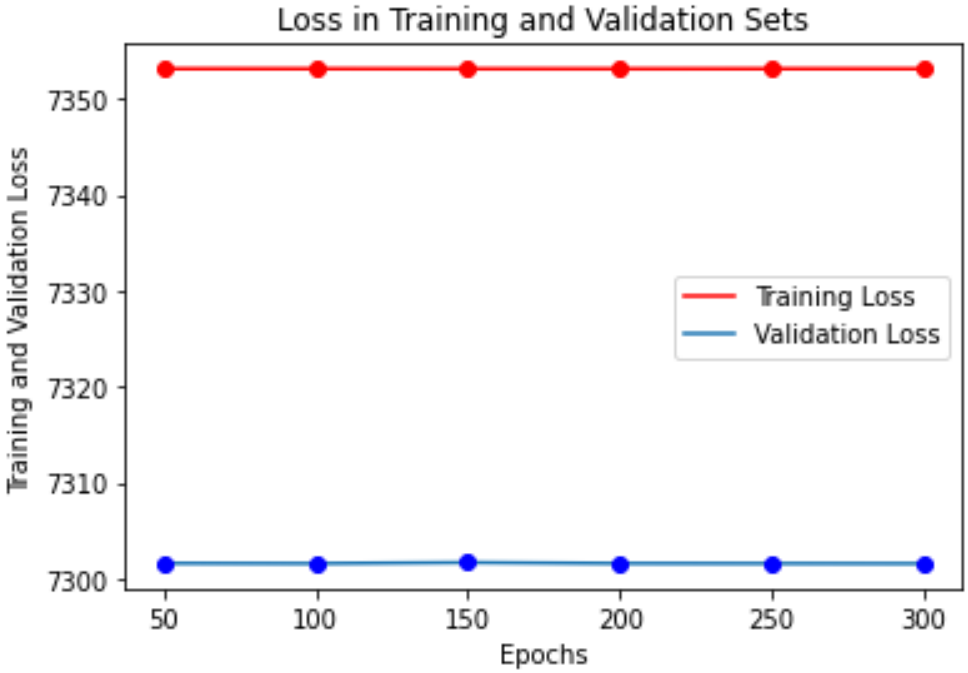
No 4: Μεγεθος Δεσμης



Εικόνα 4: Διαμόρφωση Σφάλματος για διάφορα μεγέθη δέσμης

Από το παραπάνω διάγραμμα παρατηρείται ότι το σφάλμα του validation set αυξάνεται σημαντικά, όσο αυξάνεται το μέγεθος δέσμης, ενώ το αντίστοιχο σφάλμα για το training set κυμαίνεται γύρω από ένα σταθερό εύρος τιμών.

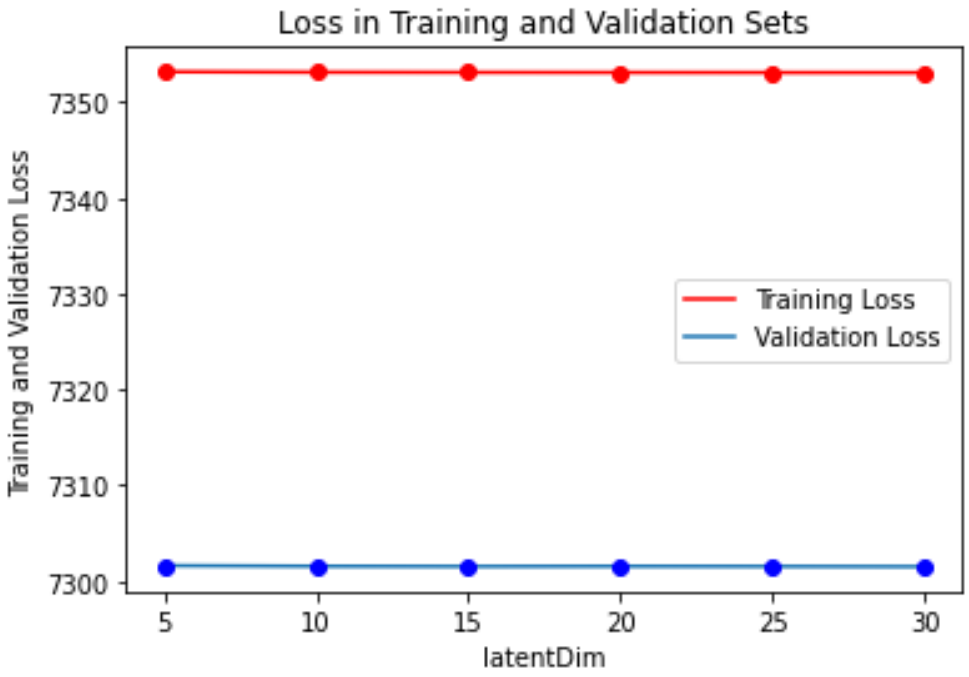
No 5: Πληθος Εποχων Εκπαιδευσης



Εικόνα 5: Διαμόρφωση Σφάλματος για διάφορες τιμές εποχών εκπαίδευσης

Όπως φαίνεται από το παραπάνω διαγράμμα, η αύξηση του πλήθους των εποχών εκπαίδευσης, δεν επηρεάζει την απόδοση του autoencoder, ως προς το σφάλμα. Παρόλα αυτά, όσο αυξάνεται το πλήθος των εποχών εκπαίδευσης, τόσο καθυστερεί η εκτέλεση του προγράμματος.

No 6: Μεγεθος Latent Dimension



Εικόνα 6: Διαμόρφωση Σφάλματος για διάφορες τιμές latent dimension

Όπως φαίνεται από το παραπάνω διάγραμμα, η αύξηση του μεγέθους του latent dimension, δεν επηρεάζει την απόδοση του autoencoder, ως προς το σφάλμα.

Ερωτημα Β:

Β1. Δομη Παραδοτεου

Το παραδοτέο για το ερώτημα 3Β αποτελείται από τα ακόλουθα αρχεία:

- search.cpp : το εκτελέσιμο αρχείο που περιλαμβάνει τη main
- classFuncs.cpp: αρχείο πηγαίου κώδικα που περιλαμβάνει τις υλοποιήσεις όλων των συναρτήσεων που αφορούν τις κλάσεις που χρησιμοποιούνται
- funcHeader.hpp: αρχεία που περιλαμβάνουν δηλώσεις των βοηθητικών συναρτήσεων που χρησιμοποιούνται για την υλοποίηση του ζητούμενου της εργασίας
- genericFuncs.cpp: αρχείο που περιλαμβάνει την υλοποίηση όλων των βοηθητικών συναρτήσεων που χρησιμοποιούνται για την υλοποίηση του ζητούμενου
- header.hpp: αρχείο με τις δηλώσεις όλων των κλάσεων που χρησιμοποιούνται
- makefile: αρχείο με τις εντολές μεταγλώττισης όλων των εκτελέσιμων της εργασίας

Στο παραδοτέο εκτός από τα προαναφερθέντα περιέχονται και δύο αρχεία, το outputDataset.txt και outputQuery.txt για την κλήση του προγράμματος.

Β2. Λειτουργικοτητα Εργασιας

Όλα τα εκτελέσιμα αρχεία που αναφέρονται παραπάνω μεταγλωττίζονται με την χρήση της εντολής:

```
make
```

Το εκτελέσιμο τρέχει με την εντολή:

```
./search -d train-images.idx3-ubyte -i outputDataset.txt -q t10k-images.idx3-
ubyte -s outputQueryset.txt -k 4 -L 3 -o output.txt
```

Με την εκκίνηση της εκτέλεσης γίνεται έλεγχος για την ορθότητα των ορισμάτων και αποδίδονται τιμές στις αντίστοιχες μεταβλητές. Στην συνέχεια, αντλούνται τα metadata, από το αρχείο που περιλαμβάνει τις εικόνες στον αρχικό χώρο, διαβάζονται μία προς μία οι εικόνες και εκτελείται ο αλγόριθμος LSH για το αρχικό σύνολο. Μόλις ολοκληρωθεί ο αλγόριθμος, διαβάζονται τα metadata και οι εικόνες του νέου διανυσματικού χώρου, καθώς και τα metadata από τα αρχεία που περιλαμβάνουν τα query sets.

Για κάθε μία από τις εικόνες λαμβάνουν χώρα οι ακόλουθοι υπολογισμοί:

- Απόσταση πλησιέστερου γείτονα στο αρχικό σύνολο και ο χρόνος εύρεσής της.
- Απόσταση πλησιέστερου γείτονα μέσω ANN από το σύνολο που παράχθηκε μέσω LSH και ο χρόνος εύρεσής της.
- Απόσταση πλησιέστερου γείτονα στον νέο περιορισμένης διάστασης διανυσματικό χώρο, απόσταση των αντίστοιχων εικόνων στους αρχικούς χώρους και ο χρόνος εύρεσής τους.
- Συνολικό άθροισμα αποστάσεων που βρέθηκαν μέσω LSH.
- Συνολικό άθροισμα αποστάσεων που βρέθηκαν μέσω Neural Net.
- Συνολικό άθροισμα πραγματικών αποστάσεων.

Σε κάθε επανάληψη τυπώνονται ο αύξοντας αριθμός του query image, οι τρεις διαφορετικές αποστάσεις καθώς και οι χρόνοι υπολογισμού τους. Μετά την ολοκλήρωση της παραπάνω επαναληπτικής διαδικασίας υπολογίζεται η μέση προσεγγιστική απόσταση ανά περίπτωση και η μέση πραγματική απόσταση από τον πλησιέστερο γείτονα. Το πηλίκο των δύο παραπάνω τελικά συνθέτει το ζητούμενο κλάσμα προσέγγισης. Τα αποτελέσματα της εκτέλεσης, τυπώνονται στο αρχείο εξόδου, όπως υποδεικνύεται στην εκφώνηση.

Ερωτημα Γ:

Γ1. Δομη Παραδοτεου

Το παραδοτέο για το ερώτημα 3Γ αποτελείται τα ακόλουθα αρχεία:

- search.cpp : το εκτελέσιμο αρχείο που περιλαμβάνει τη main
- search.py: το python αρχείο που καλείται μέσω C++

Γ2. Λειτουργικοτητα Εργασιας

Όλα το cpp εκτελέσιμο αρχείο που αναφέρεται παραπάνω μεταγλωττίζεται με την χρήση της εντολής:

```
make
```

Το εκτελέσιμο τρέχει με την εντολή:

```
./search -d train-images.idx3-ubyte -q t10k-images.idx3-ubyte -l1 train labels.idx1-ubyte -l2 t10k-labels.idx1-ubyte -o output.txt -EMD
```

Το εκτελέσιμο αυτό, εφόσον ελέγξει ότι τα ορίσματα είναι εντάξει, καλεί το αντίστοιχο εκτελέσιμο σε python. Για διευκόλυνση και ευκολότερο testing, το πρόβλημα ανάγεται σε πρόβλημα διαστάσεων 10 * 100 (10 query images, 100 train images). Εφόσον προσδιοριστούν οι εικόνες στο sets, καλείται η EMD προκειμένου να δημιουργηθεί το πρόβλημα γραμμικού προγραμματισμού. Ο στόχος είναι ο διαχωρισμός των εικόνων σε clusters και η εύρεση των μεταξύ τους αποστάσεων ως πρόβλημα LP. Αρχικά, θέλουμε να εξασφαλίσουμε ότι η συνολική φωτεινότητα μεταξύ των συγκρινόμνων εικόνων είναι ίδια, οπότε γίνεται normalize στο 1 και επιπλέον οποιαδήποτε διαφορά προκύπτει, λόγω double overflow, αφαιρείται από το 1 ο μη μηδενικό διαθέσιμο cluster.

Στη συνέχεια υπολογίζονται οι αποστάσεις μεταξύ των επιμέρους κεντροειδών σε μια εικόνα αλλά και οι ροές από κάθε cluster του query Image προς το εκάστοτε train Image, οι οποίες θεωρούνται μεταβλητές του προβλήματος και το πρόβλημα γίνεται initialize. Τέλος, λαμβάνονται υπόψιν οι περιορισμοί του δοθέντος προβλήματος με βάση τις διαφάνειες και καλείται ο solver της pulp. Εφόσον το πρόβλημα λυθεί επιτυχώς, η συνάρτηση επιστρέφει το χρόνο και τη λύση. Με τη συνάρτηση calculateCorrectness ελέγχεται η ορθότητα, όπως αυτή περιγράφεται στην εκφώνηση.

Τα αποτελέσματα για τους μέσους χρόνους και μέσα ορθά αποτελέσματα εκτυπώνονται στο output αρχείο.

Γ3. Αποτελεσματα Εκτελεσεων

Το παραπάνω αρχείο εκτελέστηκε για πειραματικούς λόγους με τις ακόλουθες παραμέτρους:

a. 16 Clusters μεγέθους 7*7, το κάθε ένα.
b. 49 Clusters μεγέθους 4*4, το κάθε ένα.

Από τα ανωτέρω πειράματα παρατηρήθηκε, ότι οι υπολογισμοί με τη μετρική Manhattan εκτελούνται σε πολύ μικρότερο χρονικό διάστημα, δεδομένου ότι η EMD, για να καταλήξει σε αποτελέσματα, πρέπει να επιλύσει ολόκληρο το πρόβλημα γραμμικού προγραμματισμού. Επίσης το μέσο πλήθος αναζητήσεων και στις δύο περιπτώσεις είναι ίδιο, ενώ η EMD παράγει αποτελέσματα με μεγαλύτερη ακρίβεια.

Και οι δύο μετρικές παρουσιάζουν 50% ακρίβεια πρόβλεψης σωστών labels.

Ερωτημα Δ:

Δ1. Δομη Παραδοτεου

Το παραδοτέο για το ερώτημα 3Δ αποτελείται από το τροποποιημένο παραδοτέο του Ερωτήματος Β της 1ης εργασίας. Η μόνη διαφορά είναι ότι πλέον μας ενδιαφέρουν 3 διαφορετικά είδη clustering, τα οποία θα χρησιμοποιήσουν τον αλγόριθμο Lloyd's και την Silhouette για την εξαγωγή αποτελεσμάτων. Πιο αναλυτικά:

- main.cpp : το εκτελέσιμο αρχείο που περιλαμβάνει τη main
- class_funtions.cpp/hpp: αρχεία που περιλαμβάνουν δηλώσεις και υλοποιήσεις όλων των συναρτήσεων
- που αφορούν τις κλάσεις που χρησιμοποιούνται
- clustering_funtions.cpp/hpp: αρχεία που περιλαμβάνουν δηλώσεις και υλοποιήσεις όλων των
- συναρτήσεων που χρησιμοποιούνται για την υλοποίηση του ζητούμενου της εργασίας
- classes.hpp: αρχείο που περιλαμβάνει τη δήλωση όλων των κλάσεων που χρησιμοποιούνται για την
- υλοποίηση του ζητούμενου
- Makefile: αρχείο με τις εντολές μεταγλώττισης όλων των εκτελέσιμων της εργασίας
- script.py: script για τη δημιουργία του clusterFile που απαιτείται με βάση τα predictions του classifier της 2ης εργασίας.

Δ2. Λειτουργικοτητα Εργασιας

Όλα τα εκτελέσιμα αρχεία που αναφέρονται παραπάνω μεταγλωττίζονται με την χρήση της εντολής:

```
make
```

Το εκτελέσιμο τρέχει με την εντολή:

```
./cluster -d train-images.idx3-ubyte -i outputDataset.txt –n clustersFile.txt -c cluster.conf -o output.txt.
```

Αρχικά, εκτελούμε το script.py προκειμένου να δημιουργήσουμε το clustersFile.txt που απαιτείται ως input για το ζητούμενο Δ. Το script διαβάζει το αρχείο που παράγει ο classifier με τα αντίστοιχα labels, και γράφει το αρχείο.

Στη συνέχεια, αφού το εκτελέσιμο ./cluster τρέξει, πραγματοποιεί αρχικά τη συσταδοποίηση Σ2 (συσταδοποίηση 1ης εργασίας) και στη συνέχεια ξεκινάει η ίδια διαδικασία για τη Σ1. Πιο αναλυτικά, οι εικόνες του νέου χώρου μπαίνουν σε ένα buffer με το οποίο καλείται η k-medians. Στην περίπτωση αυτή, όμως, εφόσον ο χώρος έχει πλέον λιγότερες διαστάσεις, τα pixels της νέας εικόνας δεν μπορούν να χρησιμοποιηθούν από την υπάρχουσα συνάρτηση. Έτσι, η συνάρτηση μεταβάλλεται και παίρνει πλέον 2 vectors ως όρισμα, έναν με τις εικόνες στο νέο χώρο και έναν με τις original εικόνες, και στις περιπτώσεις που καλείται η getVal() μιας εικόνας για κάποια τιμή pixel, το index χρησιμοποιείται για την ανάκτηση της τιμής από την original εικόνα. Προφανώς, για τη Σ2 συσταδοποίηση, ο πίνακας με τις original και τις νέες εικόνες είναι ο ίδιος.

Ο αλγόριθμος Lloyd's αντίστοιχα, παίρνει πλέον 2 τέτοια vectors και εκτελείτε ως είχε. Η Σ3 συσταδοποίηση πραγματοποιείται με διαφορετική ανάθεση clusters. Πιο συγκεκριμένα, από το clustersFile.txt, επιλέγονται οι εικόνες που αντιστοιχούν σε κάθε cluster με βάση τα εκάστοτε labels που ο classifier προέβλεψε ότι περιγράφουν. Δηλαδή. στο cluster 0 ανατίθενται οι εικόνες που έχουν προβλεφθεί ως 0, κ.ο.κ. Έτσι, η χρήση της k-medians δεν είναι πλέον απαραίτητη και η μόνη διαδικασία σε αναμονή είναι ο προσδιορισμός των κεντροειδών των clusters. Δημιουργείται λοιπόν μία νέα, παρομοια με την προηγούμενη, συνάρτηση εφαρμογής του αλγορίθμου Lloyd's, στην οποία για τα δοθέντα clusters εντοπίζονται τα νέα κέντρα.

Οι 3 αυτές συσταδοποιήσεις υπόκεινται στη διαδικασία της Silhouette, και τα αποτελέσματα γράφονται στο output file που προσδιορίζει ο χρήστης.

Με διαφορα run για διαφορετικά chunks του dataset, παρατηρούμε ότι οι τιμές που παίρνουμε από τη Silhouette κυμαίνονται κατά κύριο λόγο στο διάστημα [-1, 0] για τη Σ3, με βελτίωση των αποτελεσμάτων όσο αυξάνεται το τμήμα του dataset για το οποίο πραγματοποιείται ο έλεγχος. Αντίστοιχα, για τη Σ2 και τη Σ1 τα αποτελέσματα παρουσιάζουν μεγαλύτερη ποικιλία και οι τιμές είναι πιο συνήθως στο διάστημα [0.2, 0.4].

Τέλος υπολογίζεται και εκτυπώνεται στο αρχείο αποτελεσμάτων η τιμή της συνάρτησης στόχου, όπως υποδεικνύεται στις διαφάνειες.

Βιβλιογραφια

Η ανάπτυξη των αλγορίθμων για την υλοποίηση των ερωτημάτων της εκφώνησης βασίστηκε, κυρίως, στις διαφάνειες του μαθήματος. Συμπληρωματικό υλικό βρέθηκε στις ακόλουθες πηγές:

(1) https://wihoho.github.io/2013/08/18/EMD-Python.html
(2) https://www.coin-or.org/PuLP/CaseStudies/a_transportation_problem.html
(3) https://stackoverflow.com/questions/63986630/get-the-output-of-just-bottleneck-layer-from-autoencoder
(4) https://www.kite.com/python/answers/how-to-get-the-output-of-each-layer-of-a-keras-model-in-python
(5) https://stackoverflow.com/questions/16856788/slice-2d-array-into-smaller-2d-arrays