Αναγνώριση Προτύπων
Προγραμματιστική Εργασία 2022

Ονοματεπώνυμο: Μπαρακλιλής Ιωάννης
ΑΕΜ: 3685

Task 1

Τρόπος Εργασίας

Το task αυτό υλοποιήθηκε στο αρχείο task1.py σε Python 3 (χρησιμοποιήθηκε python 3.7 στην προκειμένη περίπτωση χρησιμοποιώντας τις βιβλιοθήκες numpy και matplotlib).

Τα αρχεία με τις εικόνες και ετικέτες του MNIST dataset βρίσκονται στον φάκελο datasets και μπορούν να διαβαστούν απευθείας από τα .py scripts που υλοποιούν τα ζητούμενα tasks.

Αρχικά, ορίζονται οι συναρτήσεις “load_labels” και “load_images” οι οποίες διαβάζουν τα δυαδικά αρχεία των ετικετών και εικόνων αντίστοιχα του dataset MNIST και τα μετασχηματίζουν σε χρηστική μορφή.
Συγκεκριμένα, στην περίπτωση της “**load_labels**”, αφού επιβεβαιωθεί η εγκυρότητα του αρχείου μέσω της επιβεβαίωσης για ταύτιση του “magic number” που βρίσκεται στα πρώτα bytes αυτού, μετά διαβάζεται το πλήθος ετικετών που περιέχει το αρχείο και τέλος διαβάζεται το σύνολο του αρχείου και επιστρέφεται λίστα με το σύνολο των ετικετών (κάθε ετικέτα είναι ένας αριθμός εντός του {0, 1, …, 255} και επομένως ενός byte επομένως αρκεί να επιστραφούν τα byte που διαβάζονται),
Όσο για τη “**load_images**”, εκείνη αφού επιβεβαιωθεί η εγκυρότητα του αρχείου μέσω της επιβεβαίωσης για ταύτιση του “magic number” που βρίσκεται στα πρώτα bytes αυτού διαβάζονται οι ιδιότητες των εικόνων (πλήθος και διαστάσεις εικόνων) και διαβάζει και αποθηκεύει τις εικόνες στις θέσεις που δόθηκαν στο όρισμα (λίστα) “target_indices” (π.χ. αν δόθηκε λίστα [1, 44, 50, 73, 100] διαβάζει τη 2η, 45η, 51, 74η και 101η και αγνοεί τις υπόλοιπες) ενώ σε περίπτωση που δόθηκε κενή λίστα, απλά διαβάζει όλες τις εικόνες. Σχετικά με το διάβασμα των εικόνων, εφόσον αυτές έχουν διαστάσεις 28×28 και οι γραμμές της εικόνας αποθηκεύονται η μία δίπλα στη άλλη στην ακολουθία bytes, απλά διαβάζεται ένα array $28\times28=274$ pixels για κάθε εικόνα που θέλουμε να διαβαστεί.

Στη συνέχεια, ορίζονται οι συναρτήσεις

- “**load_subset**” που φορτώνει το υποσύνολο των εικόνων με label που βρίσκεται στη λίστα “numbers_filter” (αν αυτό έχει τιμή “None”, φορτώνονται όλες οι εικόνες) και επιστρέφονται οι εικόνες και οι αντίστοιχες ετικέτες κάθε εικόνας (σε αντιστοιχία θέσεων στις λίστες) και
- “**get_M_N_Ltr_Lte**” που φορτώνει και επιστρέφει τους ζητούμενους (από την εκφώνηση) πίνακες $\boldsymbol{M}, \boldsymbol{N}, \boldsymbol{L}_{tr}, \boldsymbol{L}_{te}$.

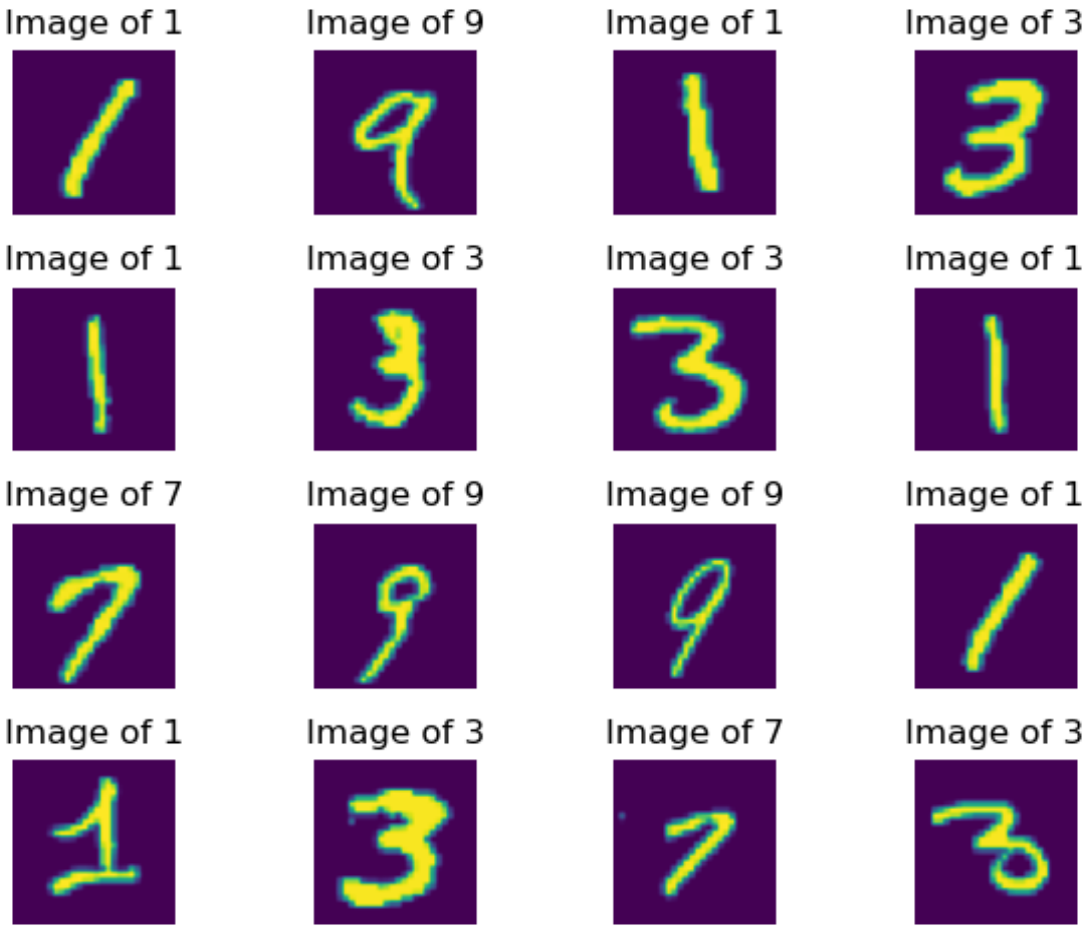
Τέλος, υπάρχει ένα εκτελέσιμο τμήμα το οποίο επιδεικνύει το ότι η διαδικασία φόρτωσης εικόνων έγινε σωστά, όπου εμφανίζονται οι διαστάσεις των πινάκων και δημιουργείται ένα γράφημα που οπτικοποιεί 16 εικόνες.

Αποτελέσματα

Αν εκτελέσουμε τον κώδικα, θα εμφανιστεί το ακόλουθο, στο το οποίο φαίνονται οι διαστάσεις των πινάκων που φαίνεται να συμβαδίζουν με εκείνες της εκφώνησης:

```
M dimensions: (25087, 784)
N dimensions: (4182, 784)
L_tr dimensions: (25087,)
L_te dimensions: (4182,)
```

Μετά παράγεται το ακόλουθο “γράφημα” που περιέχει τις εικόνες όπου μπορούμε να δούμε πως η διαδικασία φόρτωσης έγινε σωστά:



Επιπλέον, στα επόμενα tasks το μόνο που χρειάζεται να γίνει για φόρτωση εικόνων είναι η κλήση της συνάρτησης “get_M_N_Ltr_Lte” του task1.py.

Task 2

Τρόπος Εργασίας

Το task αυτό υλοποιήθηκε στο αρχείο task2.py σε Python 3 (χρησιμοποιήθηκε python 3.7 στην προκειμένη περίπτωση χρησιμοποιώντας τις βιβλιοθήκες numpy και matplotlib).

Η αναδιαμόρφωση των εικόνων σε τετράγωνους πίνακες 28×28 και η εξαγωγή δισδιάστατου διανύσματος για κάθε εικόνα (βάσει της διαδικασίας που περιγράφεται στην εκφώνηση) γίνεται μέσω της συνάρτησης "**reshape_and_extract_feature_vector**" η οποία επιστρέφει τον ζητούμενο πίνακα $\hat{\boldsymbol{M}}$.

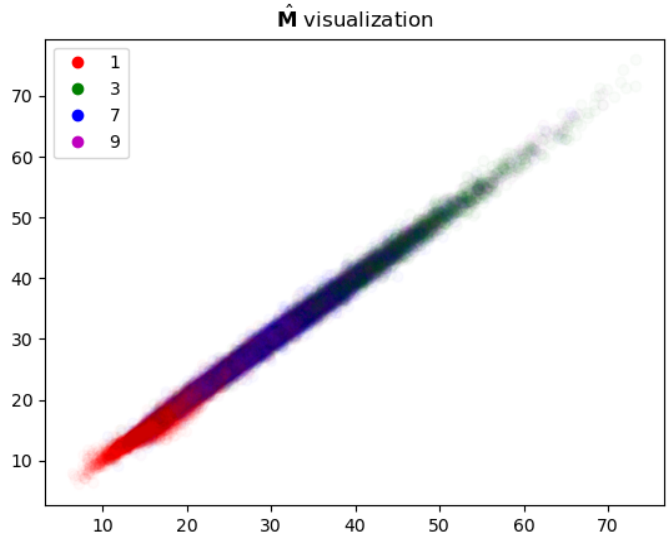
Η δημιουργία του scatter plot (με πολύ χαμηλή διαφάνεια για να φαίνονται οι επικαλύπτουσες κλάσεις) για την οπτικοποίηση των γραμμών του $\hat{\boldsymbol{M}}$ γίνεται από τη συνάρτηση "**visualize_scatter**" με χρώματα:

- Κόκκινο για την κλάση 1,
- Πράσινο για την κλάση 3,
- Μπλε για την κλάση 7 και
- Ματζέντα για την κλάση 9.

Τέλος, υπάρχει ένα εκτελέσιμο τμήμα το οποίο εκτελεί απλά τη visualize_scatter και εκτυπώνεται το ζητούμενο γράφημα.

Αποτελέσματα

Αν εκτελέσουμε τον κώδικα, θα εμφανιστεί το ακόλουθο γράφημα οπτικοποίησης του $\hat{\boldsymbol{M}}$:



Σε αυτό το γράφημα μπορούμε να παρατηρήσουμε πως οι δισδιάστατες απεικονίσεις των κλάσεων συμπίπτουν σε μεγάλο βαθμό και είναι δύσκολος (έως αδύνατος) ο διαχωρισμός τους.

Παρ’ όλα αυτά, λόγο έντασης χρωμάτων φαίνεται πως η κλάση 1 βρίσκεται στο “πρώτο” τμήμα (κοντά στο σημείο (15,15)) του σχήματος (αν το θεωρήσουμε ότι εκτείνεται από κάτω αριστερά προς τα πάνω δεξιά), η κλάση 9 βρίσκεται στο “δεύτερο” τμήμα (κοντά στο σημείο (26,26)) του σχήματος, η κλάση 7 βρίσκεται στο “τρίτο” τμήμα (κοντά στο σημείο (37,37)) του σχήματος και η κλάση 3 βρίσκεται στο “τέταρτο” τμήμα (κοντά στο σημείο (50, 50)) του σχήματος.

Task 3

Τρόπος Εργασίας

Το task αυτό υλοποιήθηκε στο αρχείο task3.py σε Python 3 (χρησιμοποιήθηκε python 3.7 στην προκειμένη περίπτωση χρησιμοποιώντας τις βιβλιοθήκες numpy και matplotlib).

Η υλοποίηση του αλγορίθμου Maximin (με βάση το πρότυπο του αλγορίθμου που είδαμε στο φροντιστηριακό μάθημα) που χρησιμοποιείται στην αρχική ανάθεση κέντρων ομάδων γίνεται από τη συνάρτηση "**maximin**" η οποία έχει σαν συνθήκη τερματισμού την επίτευξη επιθυμητού αριθμού κέντρων (στην προκειμένη περίπτωση 4, όπως ζητείται).

Η υλοποίηση του αλγορίθμου K-Means (με βάση το πρότυπο του αλγορίθμου που είδαμε στο φροντιστηριακό μάθημα) που χρησιμοποιεί για αρχική ανάθεση κέντρων ομάδων τον αλγόριθμο Maximin γίνεται από τη συνάρτηση "**k_means**".

Ο τρόπος υπολογισμού των αποστάσεων μεταξύ δύο σημείων, για τους παραπάνω δύο αλγορίθμους, δίνεται ως παράμετρος και ο προκαθορισμένος τρόπος (που χρησιμοποιείται εδώ) είναι η ευκλείδεια απόσταση.

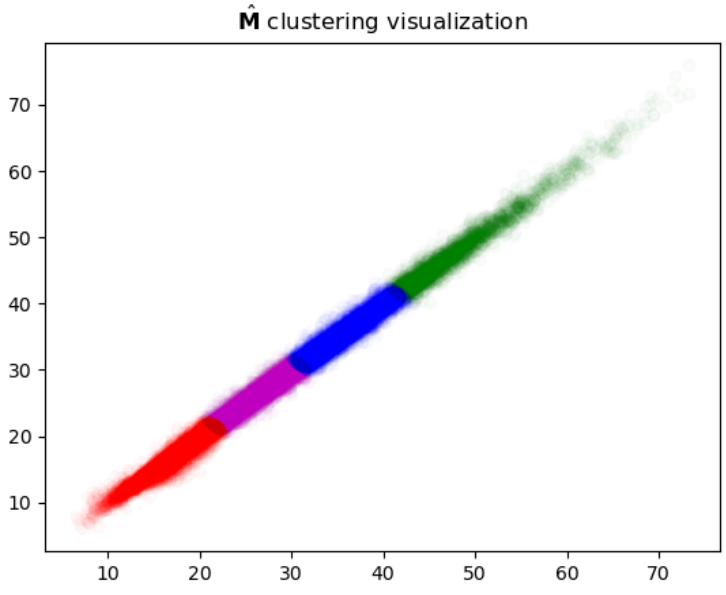
Η οπτικοποίηση του αποτελέσματος της ομαδοποίησης του $\hat{M}$ (όπως υπολογίστηκε στο task 2) σε scatter plot γίνεται μέσω της συνάρτησης "**visualize_clustering**".

Ο υπολογισμός του clustering purity (ο τύπος που χρησιμοποιήθηκε βρέθηκε εδώ) κάθε ομάδας γίνεται μέσω της συνάρτησης "**clustering_purity**".

Τέλος, υπάρχει ένα εκτελέσιμο τμήμα το οποίο μετά την εκτέλεση της ομαδοποίησης στο $\hat{M}$ καλεί τη visualize_clustering για να εκτυπωθεί το ζητούμενο γράφημα (ο χρωματισμός έγινε έτσι ώστε οι αντίστοιχες περιοχές με εκείνες στο γράφημα του task 2 να χρωματίζονται με το ίδιο χρώμα) και υπολογίζεται και εκτυπώνεται το clustering purity της προηγούμενης ομαδοποίησης.

Αποτελέσματα

Αν εκτελέσουμε τον κώδικα, θα εμφανιστεί το ακόλουθο γράφημα οπτικοποίησης της ομαδοποίησης:



Όπως βλέπουμε, οι ομάδες (που βρίσκονται σε αντίστοιχες περιοχές με αυτές του task2) που φαίνονται στο γράφημα είναι πολύ πιο ξεκάθαρες και χωρίζουν τον χώρο πολύ καλύτερα από ότι στην οπτικοποίηση χωρίς ομαδοποίηση και βάσει του purity (που υπολογίζεται στη συνέχεια) $\sim 46.85\%$ θα λέγαμε ότι χωρίζει τον χώρο σε διαφορετικές κλάσεις όχι ιδιαίτερα καλά αλλά τουλάχιστον δημιουργείται κάποια οπτικοποίηση της κατανομής των δεδομένων καλύτερη από την απλή (του task2).

Επίσης, θα εμφανιστεί και το αποτέλεσμα του clustering purity που είναι (με στρογγυλοποίηση στα 4 δεκαδικά) $\sim 0.4685 = 46.85\%$:

```
Purity = 0.4685295172798661
```

Άρα, το $\sim 46.85\%$ των σημείων του συνόλου εκπαίδευσης ανήκουν σε ομάδα όπου η πλειοψηφία ανήκει στην ίδια κλάση, δηλαδή έχουν αντιστοιχηθεί στη “σωστή” ομάδα και έτσι είναι ομαδοποίηση η οποία δεν είναι ιδιαίτερα ακριβής με ομάδες που δεν είναι ιδιαίτερα αμιγείς με στοιχεία της ίδιας κλάσης.

Επομένως, αυτή η μέθοδος εξαγωγής διανυσμάτων δύο διαστάσεων δεν είναι ιδιαίτερα ακριβής.

Task 4

Τρόπος Εργασίας

Το task αυτό υλοποιήθηκε στο αρχείο task4.py σε Python 3 (χρησιμοποιήθηκε python 3.7 στην προκειμένη περίπτωση χρησιμοποιώντας τις βιβλιοθήκες numpy, matplotlib και scipy).

Η Principal Component Analysis (PCA) (στον ορισμό που χρησιμοποιήθηκε για την εφαρμογή της εδώ) ορίζεται ως η ορθογώνια προβολή των δεδομένων σε γραμμικό χώρο λιγότερων διαστάσεων έτσι ώστε να μεγιστοποιείται η διακύμανση των προβαλλόμενων δεδομένων.

Επομένως, αρκεί να βρεθεί η βάση του χώρου και να γίνει προβολή των δεδομένων σε αυτή. Αυτά, υλοποιούνται από τις συναρτήσεις:

- "**pca**", που υπολογίζει τη βάση του χώρου ως το τον πίνακα που συντίθεται από τα ιδιοδιανύσματα που αντιστοιχούν στις V (νέος αριθμός διαστάσεων) μεγαλύτερες ιδιοτιμές του πίνακα συνδιακύμανσης.
- "**transform**", μετασχηματίζει τα δεδομένα σε αντίστοιχα μικρότερης διάστασης μέσω κάποιας βάσης που υπολογίστηκε βάσει της "pca".

Τα παραπάνω αιτιολογούνται και βασίζονται στη θεωρία από το βιβλίο [ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ ΚΑΙ ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ, 2^{η} έκδοση, Christopher M. Bishop, σελίδες 621-623].

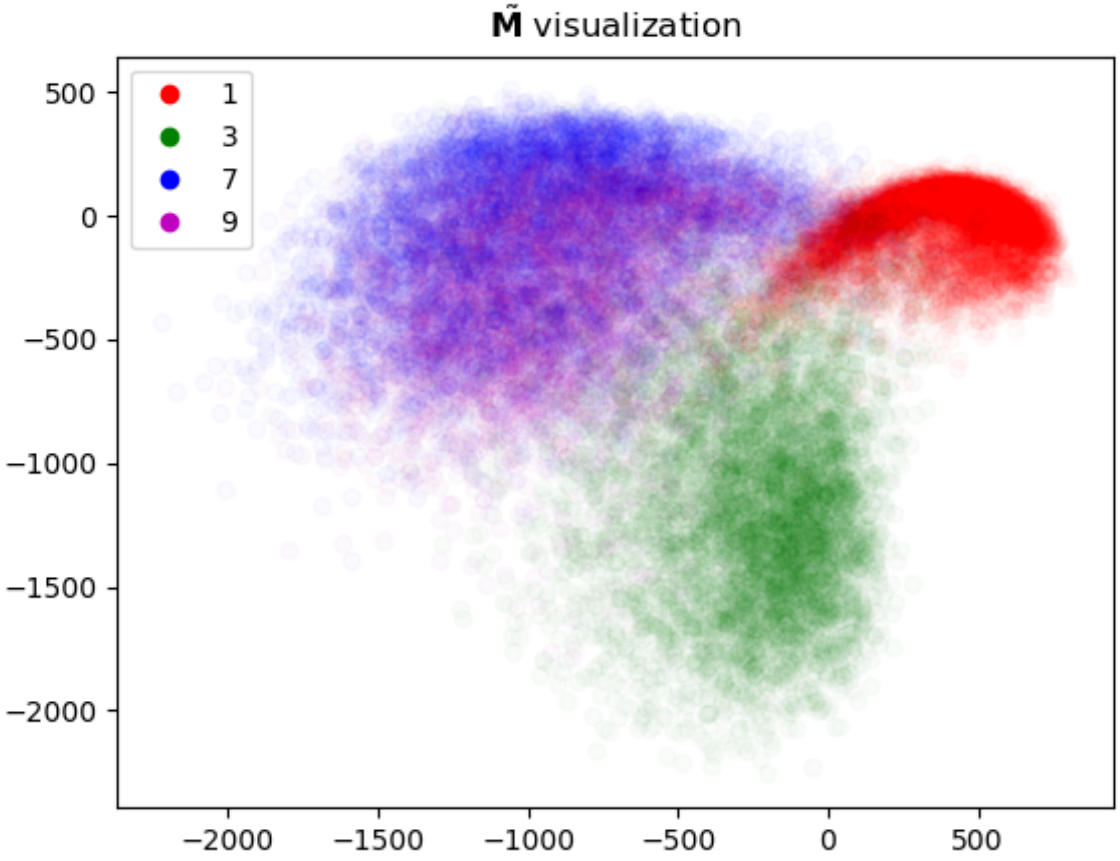
Επίσης, ορίζεται η συνάρτηση "**visualize_plain_scatter**" που παράγει την οπτικοποίηση με scatter plot ενός πίνακα M και χρώματα βάσει ετικετών L_tr.

Τέλος, υπάρχει ένα εκτελέσιμο τμήμα το οποίο υλοποιεί τα ζητούμενα της εκφώνησης:

- Εφαρμόζει την PCA για τη μείωση διαστάσεων του αρχικού πίνακα $\boldsymbol{M}$ και δημιουργία του πίνακα $\widetilde{\boldsymbol{M}}$ (για νέες διαστάσεις $V = 2, 25, 50, 100$ σε αντίστοιχα σημεία).
- Μετασχηματίζει τα δεδομένα με προβολή σε χώρους μικρότερων διαστάσεων.
- Δημιουργεί το scatter plot (στην περίπτωση V = 2) για τον $\widetilde{\boldsymbol{M}}$.
- Εκτελεί ομαδοποίηση (με την k_means με αρχικοποίηση Maximin του task 3) στα στοιχεία του $\widetilde{\boldsymbol{M}}$ και δημιουργεί scatter plot για τα αποτελέσματα αυτού.
- Για τα $V = 2, 25, 50, 100$ εκτελεί ομαδοποίηση, υπολογίζει clustering purity για κάθε ομάδα και βρίσκει το V_{max} που το μεγιστοποιεί και το εμφανίζει.

Αποτελέσματα

Αν εκτελέσουμε τον κώδικα, θα εμφανιστούν τα ακόλουθα γραφήματα (scatter) για την οπτικοποίηση του $\widetilde{\boldsymbol{M}}$ και το αποτέλεσμα ομαδοποίησης αυτού:



Οπτικοποίηση του $\widetilde{\boldsymbol{M}}$

Οπτικοποίηση της ομαδοποίησης του $\widetilde{\boldsymbol{M}}$

Βλέπουμε ότι στην οπτικοποίηση του $\widetilde{\boldsymbol{M}}$ οι ομάδες χωρίζουν τις κλάσεις στις οποίες ανήκουν οι εικόνες πολύ καλύτερα από την περίπτωση χωρίς PCA (task 2 και 3) όπου μπορεί εύκολα κάποιος ότι η κλάση με τα ψηφία 0 “βρίσκεται” στην περιοχή πάνω δεξιά (κοντά στο σημείο (450, 0)) και η κλάση με τα ψηφία 1 “βρίσκεται” στην περιοχή κάτω δεξιά (κοντά στο σημείο (0, -1250)). Όσο για τις κλάσεις με ψηφία 7 και 9 η διάκριση είναι πολύ πιο δύσκολη. Ωστόσο, λόγο έντασης χρωμάτων φαίνεται πως η κλάση 7 βρίσκεται κοντά στο σημείο (-900, 300) και η 9 λίγο πιο κάτω.

Μετά το clustering, οι ομάδες είναι πολύ πιο ξεκάθαρες και χωρίζουν τον χώρο ομοιόμορφα και βάσει του purity (που υπολογίζεται στη συνέχεια) $\sim 72.24\,\%$ θα λέγαμε ότι χωρίζει τον χώρο σε διαφορετικές κλάσεις αρκετά καλά και ασφαλώς πολύ καλύτερα από την απλοϊκή εξαγωγή δισδιάστατων σημείων των task 2 και 3.

Επίσης, θα εμφανιστούν και τα αποτελέσματα του clustering purity κάθε ομαδοποίησης για $V = 2, 25, 50, 100$ που είναι (με στρογγυλοποίηση στα 4 δεκαδικά):

- Για V = 2: $\sim 0.7224 = 72.24\,\%$,
- Για V = 25: $\sim 0.9238 = 92.38\,\%$,
- Για V = 50: $\sim 0.9325 = 93.25\,\%$,
- Για V = 100: $\sim 0.8785 = 87.85\,\%$.

Επομένως, προκύπτει πως $V_{max} = 50$, όπου το $\sim 93.25\,\%$ των σημείων του συνόλου εκπαίδευσης ανήκουν σε ομάδα όπου η πλειοψηφία ανήκει στην ίδια κλάση, δηλαδή έχουν αντιστοιχηθεί στη “σωστή” ομάδα και έτσι είναι η καλύτερη ομαδοποίηση.

Αυτά φαίνονται και στα αποτελέσματα εκτέλεσης του προγράμματος:

```
Purity with V = 2:  0.7224458883086857
Purity with V = 25:  0.9238649499740902
Purity with V = 50:  0.9325945708932913
Purity with V = 100:  0.878582532785905

Result: Vmax =  50
```

Task 5

Τρόπος Εργασίας

Το task αυτό υλοποιήθηκε στο αρχείο task5.py σε Python 3 (χρησιμοποιήθηκε python 3.7 στην προκειμένη περίπτωση χρησιμοποιώντας τη βιβλιοθήκη numpy).

Ο γκαουσιανός Naive Bayes ταξινομητής εκπαιδεύεται μέσω της συνάρτησης "**train_naive_bayes**" όπου υπολογίζονται οι κανονικές κατανομές (παράμετροι βάσει μέγιστης πιθανοφάνειας βάσει θεωρίας ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ ΚΑΙ ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ, 2η έκδοση, Christopher M. Bishop], σελίδες 112-113) των χαρακτηριστικών (που υποθέτουμε πως έχουν, και στον Naive Bayes ταξινομητή θεωρούνται ανεξάρτητες) βάσει δεδομένης κλάσης για κάθε κλάση και επιστρέφεται ο ταξινομητής που ορίζεται από τη συνάρτηση "**gausian_naive_bayes_classification**" δυναμικά η οποία χρησιμοποιεί τις παραπάνω υπολογισμένες κατανομές για να ταξινομήσει ένα άγνωστο δείγμα $\boldsymbol{x}$ σε μία από τις 4 κλάσεις στις οποίες εκπαιδεύτηκε όπου ταξινομεί ένα δείγμα στην κλάση με μεγαλύτερη πιθανοφάνεια (βάσει της σχέσης από [Αναγνώριση Προτύπων, S. Theodoridis, K. Koutroubas, 4η έκδοση], σελίδα 70).

Τέλος, υπάρχει ένα εκτελέσιμο τμήμα το οποίο αφού μετασχηματίσει τα δεδομένα με την PCA (από συναρτήσεις του task 4) για $V = V_{max} = 50$ παράγει τους $\widetilde{\boldsymbol{M}}, \widetilde{\boldsymbol{N}}$, με τους μετασχηματισμένους πίνακες εικόνων για τα δεδομένα εκπαίδευσης και ελέγχου.
Στη συνέχεια, εκπαιδεύει το μοντέλο και το χρησιμοποιεί για να υπολογίσει την ορθότητα (accuracy) του classification βάσει των $\widetilde{\boldsymbol{N}}, \boldsymbol{L}_{te}$ που ορίζεται ως ο λόγος των ορθών αποτελεσμάτων ταξινόμησης προς το σύνολο των στοιχείων ελέγχου.

Αποτελέσματα

Αν εκτελέσουμε τον κώδικα, θα εκτυπωθεί η ορθότητα του classification του συνόλου δοκιμών που προκύπτει (με στρογγυλοποίηση στα 4 δεκαδικά) $\sim 0.9273 = 92.73\,\%$ η οποία είναι αρκετά καλή. Αυτό, φαίνεται και στο αποτέλεσμα εκτέλεσης:

```
Test set classification accuracy: 0.9273075083692014
```