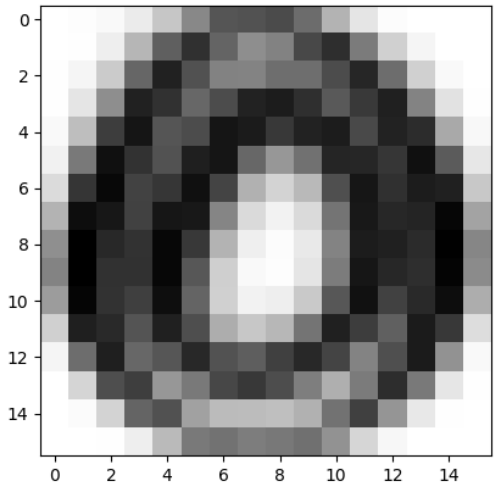
Το επόμενο βήμα (**Βήμα 8**) είναι να σχεδιάσουμε ένα από τα ψηφία που επιθυμούμε χρησιμοποιώντας όχι την μέση τιμή, αλλά την διασπορά των pixel των εικονων που αντιπροσωπεύουν το επιθυμητό ψηφίο. Αυτό γίνεται με την συνάρτηση $plot\_digit\_variance$. Για παράδειγμα, για το ψηφίο 0 παίρνουμε την εξής εικόνα:

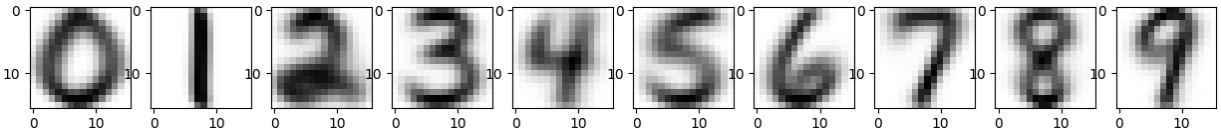


Εικόνα 1: Απεικόνιση του ψηφίου 0 μέσω της διασποράς των pixel

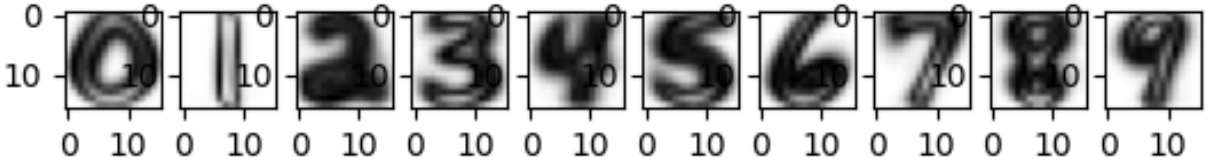
Αν συγκρίνουμε την εικόνα που πήραμε από το ψηφίο 0 μέσω της μέσης τιμής με αυτήν που πήραμε μέσω τις διασποράς θα παρατηρήσουμε κάποιες σχετικά αναμενόμενες διαφορές. Η εικόνα που αντιστοιχεί στην διασπορά έχει τα πιο σκούρα της pixels αυτά που τα αντίστοιχα στην εικόνα της μέσης τιμής είναι τα λιγότερο σκούρα. Για παράδειγμα τα pixels της εικόνας της μέσης τιμής για το 0 είναι περισσότερο σκούρα στις δύο "κορυφές" του 0. Αντίστοιχα, τα pixels της εικόνας της διασποράς είναι λιγότερο σκούρα (περισσότερο γκρί παρά μαύρα) στις κορυφές αυτές. Αυτό είναι αναμενόμενο, καθώς η διασπορά είναι στην ουσία η απόσταση μιας μεταβλητής από την μέση τιμή της.

Για το **Βήμα 9** υπολογίζουμε την μέση τιμή και την διασπορά όλων των pixel για όλα τα δεδομένα εκπαίδευσης. Αυτό μπορεί να γίνει πολύ απλά χρησιμοποιωντας list-comprehension της python και τις συναρτήσεις $digit\_mean$ και $digit\_variance$ αντίστοιχα. Παίρνουμε έτσι δύο λίστες 10 γραμμών και 256 στηλών ($mean\_X$ και var_X). Κάθε γραμμή αντιστοιχεί σε ένα ψηφίο και κάθε στήλη στην μέση τιμή και την διασπορά του εκάστοτε pixel.

Για να σχεδιάσουμε όλα τα ψηφία χρησιμοποιούμε τις συνάρτησεις $plot\_digits\_by\_mean$ και $plot\_digits\_by\_variance$, οι οποίες χρησιμοποιούν 10 φορές η καθε μία τις συναρτήσεις $digit\_mean$ και $digit\_variance$ αντίστοιχα.



Εικόνα 2: Απεικόνιση όλων των ψηφίων μέσω της μέσης τιμής των pixel



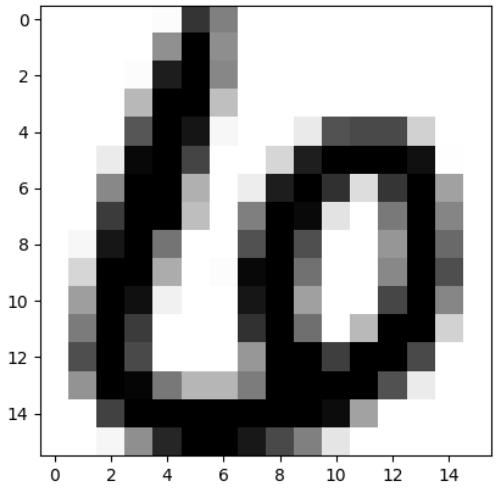
Εικόνα 3: Απεικόνιση όλων των ψηφίων μέσω της διασποράς των pixel

Από το **Βήμα 10** και μετά ξεκινούν οι δοκιμές του ταξινομητή της Ευκλείδιας απόστασης. Αυτός υλοποιείται με την βοήθεια δύο συναρτήσεων. Η συνάρτηση $euclidean\_distance$ υπολογίζει την Ευκλείδια απόσταση μεταξύ δύο διανυσμάτων (ή πινάκων) μέσω της σχέσης:

$$d(A,B) = \sqrt{(A[1]-B[1])^2 + (A[2]-B[2])^2 + \cdots + (A[N]-B[N])^2} \qquad (1)$$

Στην συνέχεια για την ταξινόμηση των στοιχείων του test set καλούμε την συνάρτηση $euclidean\_distance\_classifier$. Η συνάρτηση αυτή παίρνει ως ορίσματα τα χαρακτηριστικά του test set (test features) καθώς και την λίστα που περιλαμβάνει τις μέσες τιμές όλων των ψηφίων του training set. Υπολογίζει για το κάθε στοιχείο του test set τις ευκλείδιες αποστάσεις του στοιχείου αυτού από τις μέσες τιμές. Επιλέγεται η μικρότερη από αυτές, η οποία ελάχιστη απόσταση είναι αντιπροσωπευτική της κλάσης στην οποία ταξινομήθηκε τελικά το στοιχείο που επιλεξαμε.

Ως πρώτη δοκιμή του ταξινομητή, διαλέγουμε το στοιχείο υπ'αριθμόν 101 των test δεδομένων και το δίνουμε ως είσοδο στον ταξινομητή μας. Για να έχουμε μία καλύτερη εικόνα, το στοιχείο αυτό είναι το εξής:



Εικόνα 4: Το ψηφίο υπ'αριθμόν 101 των test δεδομένων

Αν τρέξουμε τον ταξινομητή μας θα διαπιστώσουμε ότι η ταξινόμηση δεν είναι σωστή. Όπως μπορούμε να διακρίνουμε στην παραπάνω εικόνα, το ψηφίο που αναπαριστά είναι το 6. Παρόλα αυτά ο ταξινομητής δίνει την πρόβλεψη 0. Οπότε "μπερδεύει"το παραπάνω εξάρι με μηδενικό. Αυτό δεν πρέπει να μας αποθαρρύνει αφού ένα μόνο test δεδομένο δεν αρκεί για να βγάλουμε συμπέρασμα για τον ταξινομητή μας. Εξάλλου η παραπάνω εικόνα είναι πράγματι ένα ασυνήθιστο 6 για τα τυπικά χειρόγραφα δεδομένα.

Ερχόμαστε τώρα στο **Βήμα 11**, όπου τροφοδοτούμε τον ταξινομητή μας με όλα τα test δεδομένα. Η συνάρτηση $euclidean\_distance\_classifier$ θα μας γυρίσει ένα % ποσοστό επιτυχίας (δηλαδή τον αριθμό των δεδομένων που ταξινομήθηκαν επιτυχώς διαιρεμένο με τον συνολικό αριθμό των δεδομένων) και μία λίστα με όλες τις προβλέψεις του ταξινομητή (δηλαδή μία λίστα που θα μας λέει σε ποια κατηγορία ταξινομήθηκε το κάθε test δεδομένο). Αν τρέξουμε τον κώδικα και τυπώσουμε τις τιμές τότε θα λάβουμε στην οθόνη μας:

The accuracy of the euclidean distance classifier is: 81.42 %

Το ποσοστό αυτό είναι αρκετά ικανοποιητικό για τον περιορισμένο αυτό όγκο δεδομένων που είχαμε στην διάθεσή μας.

Στο **Βήμα 12** υλοποιούμε τον ταξινομητή της Ευκλείδιας απόστασης σαν έναν scikit-learn estimator. Ουσιαστικά δημιουργούμε μία κλάση EuclideanDistanceClassifier της python, μέσα στην οποία θα καλούμε τις κατάλληλες συναρτήσεις που έχουμε ήδη σχηματίσει προηγουμένως στον κώδικα. Η μόνη μεταβλητή που θα αρχικοποιείται στην κλάση είναι η μέση τιμή των κατηγοριών (των ψηφίων) και αυτή θα υπολογίζεται στην συνέχεια από την συνάρτηση fit μέσω των train δεδομένων. Η συνάρτηση predict θα έχει ως είσοδο τα test

features και θα επιστρέφει την λίστα με τις προβλέψεις του ταξινομητή, ενώ η συνάρτηση score θα έχει ως είσοδο τα test features και τα test labels επιστρέφει το ποσοστό επιτυχίας.

Αν θέλουμε να καλέσουμε τώρα τον ταξινομητή μας σαν να ήταν ταξινομητής του scikit-learn θα πρέπει απλά να καλέσουμε την κλάση μας ώς:

clf = EuclideanDistanceClassifier()

Στην συνέχεια θα πρέπει να τροφοδοτήσουμε τα train δεδομένα στην κλάση για να υπολογίσουμε τις μέσες τιμές, μέσω της συνάρτησεις fit:

clf.fit(out training features)

Τέλος, αν θέλουμε να μας επιστραφούν οι προβλέψεις ή η ακρίβεια του ταξινομητή καλούμε τις συναρτήσεις predict ή score αντίστοιχα:

clf.predict(our test features)
clf.score(our test features, our test labels)

Προφανώς αν τυπώσουμε το score μέσω της παραπάνω κλάσης θα λάβουμε το ίδιο ακριβώς αποτέλεσμα με την συνάρτηση $euclidean\_distance\_classifier$.

Στο **Βήμα 13** εξερευνούμε λίγο παραπάνω τον ταξινομητή που δημιουργήσαμε υπολογίζοντας το cross-validation-score με την χρήση της τεχνικής K-Folding καθώς επίσης σχεδιάζουμε την περιοχή απόφασης και την καμπύλη εκμάθησης του ταξινομητή.

Για τον υπολογισμό του cross-validation-score αντλούμε από το scikit-learn τις κλάσεις KFOLD και $cross\_val\_score$. Στην συνέχεια καλούμε την συνάρτηση k_fold_cv η οποία παίρνει ως είσοδο τα train features και τα train labels και τυπώνει το cross-validation-score και το cross-validation-error. Αναμένουμε σαν cross-validation-score να λάβουμε κάτι λίγο παραπάνω από το score του ταξινομητή μας. Πράγματι, αν καλέσουμε την συνάρτηση τότε θα λάβουμε στην οθόνη:

The 5-fold cross-validation score is 84.858036 +-0.181618
The 5-fold cross-validation error is 15.141964 +-0.181618

Το ποσοστό του cross-validation είναι περίπου 85 %, λίγο μεγαλύτερο από το 82 % του ταξινομητή.

Τέλος σχεδιάζουμε την καμπύλη εκμάθησης του ταξινομητή μας. Αυτό γίνεται μέσω της συνάρτησης $plot\_learning\_curve$. Παίρνει ως είσοδο τον ταξινομητή (την κλάση που τον αντιπροσωπεύει) και όλα τα train δεδομένα (features, labels).

Εικόνα 5: Η καμπύλη εκμάθησης του Ευκλείδιου ταξινομητή

Όπως αναμέναμε, το cross-validation-score αρχικά αυξάνεται και, όσο προχωρά η εκπαίδευση, τείνει να εξισωθεί με το training score.