Εθνικό Μετσόβιο Πολυτεχνείο
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Αναγνώριση Προτύπων

Ροή Σ: Σήματα, Έλεγχος, Ρομποτική
9ο Εξάμηνο

1η Εργαστηριακή Άσκηση:
Οπτική Αναγνώριση Ψηφίων

Χρήστος Δημόπουλος - 03117037
chrisdim1999@gmail.com

Οκτώβριος, 2021

Contents

Θέμα Εργαστηριακής Άσκησης

Σκοπός της πρώτης εργαστηριακής άσκησης είναι η υλοποίηση ενός συστήματος οπτικής αναγνώρισης ψηφίων. Τα δεδομένα προέρχονται από την US Postal Service (γραμμένα στο χέρι σε ταχυδρομικούς φακέλους και σκαναρισμένα) και περιέχουν τα ψηφία από το 0 έως το 9 και διακρίνονται σε train και test.
Ως προπαρασκευαστικό στάδιο, αρχικά εφαρμόζουμε **preprocessing** στα δεδομένα μας και παρουσιάζουμε ορισμένα στατιστικά χαρακτηριστικά τους (μέση τιμή και διασπορά). Στη συνέχεια, υλοποιούμε έναν **ταξινόμητη Ευκλείδιας Απόστασης** και αξιολογούμε τις επιδόσεις του στο εν λόγω task.
Στο κυρίως μέρος της άσκησης, υλοποιούμε έναν **Naive Bayes Classifier** και συγκρίνουμε την επίδοση του με αυτές έτοιμων ταξινομητών της βιβλιοθήκης scikit-learn. Έπειτα, πειραματιζόμαστε με τεχνικές **ensembling** και **boosting** με στόχο την επίτευξη καλύτερων αποτελεσμάτων ταξινόμησης. Τέλος, προσεγγίζουμε το εν λόγω πρόβλημα ταξινόμησης με χρήση **Νευρωνικών Δικτύων**.

Βήματα Προπαρασκευής

1. Ανάγνωση Δεδομένων

Αρχικά, διαβάζουμε τα δεδομένα εισόδου από τα αρχεία *train.txt* και *test.txt*. Τα δεδομένα διαβάζονται σε μορφή συμβατή με το scikit-learn σε 4 πίνακες `X_train`, `X_test`, `y_train` και `y_test`. Ο πίνακας `X_train` περιέχει τα δείγματα εκπαίδευσης, χωρίς τα labels και είναι διάστασης `(n_samples_train x n_features)`. Ο `y_train` είναι ένας μονοδιάστατος πίνακας μήκους `n_samples` και περιέχει τα αντίστοιχα labels για τον `X_train`. Αντίστοιχα για τα test δεδομένα.

2. Απεικόνιση Ψηφίου υπ' αριθμόν 131

Ακολούθως, απεικονίζουμε το υπ' αριθμόν ψηφίο 131 από το train dataset. Το απεικονιζόμενο ψηφίο είναι το *'9'*, όπως ήδη γνωρίζουμε από το label του.

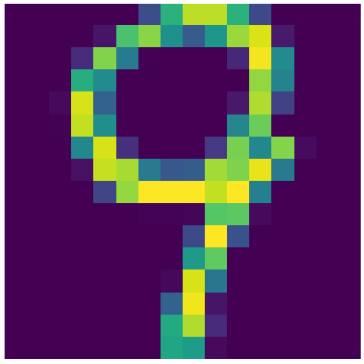


Figure 1: Ψηφίο 131 του Training Set.

.

3. Απεικόνιση Ψηφίων 0-9

Ομοίως, απεικονίζουμε τυχαία δείγματα για κάθε ένα εκ των ψηφίων 0-9, τα οποία αντιστοιχούν στις 10 κλάσεις του προβλήματος κατηγοριοποίησης. Τα δείγματα αυτά φαίνονται στο Σχήμα 2.

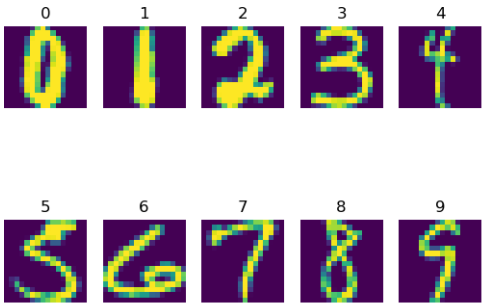


Figure 2: Τυχαία Δείγματα των Ψηφίων 0-9.

.

4. Μέση Τιμή pixel (10,10) για το Ψηφίο 0

Στη συνέχεια, εστιάζουμε στις στατιστικές ιδιότητες των χαρακτηριστικών των προς κατηγοριοποίηση ψηφίων. Αρχικά, προσδιορίζουμε τη μέση τιμή για το pixel (10,10) μόνο των ψηφίων με ένδειξη 0 από το train dataset:

Mean Value of pixel (10,10) for Digits 0: ***-0.504***

5. Διασπορά pixel (10,10) για το Ψηφίο 0

Ομοίως, προσδιορίζουμε τη διασπορά για το pixel (10,10) μόνο των ψηφίων με ένδειξη 0 από το training dataset:

Variance of pixel (10,10) for Digits 0: ***0.524***

6. Μέση Τιμή & Διασπορά για κάθε pixel των Ψηφίων 0

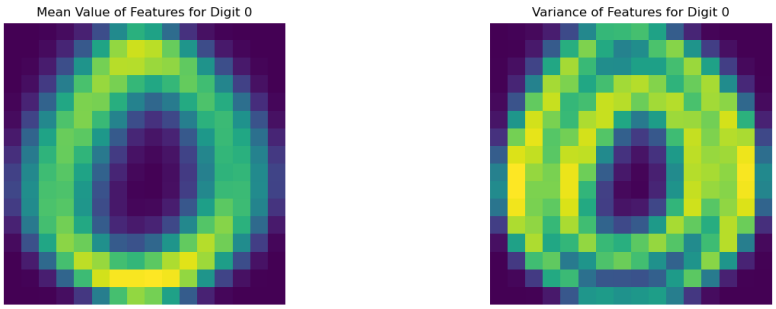
Στο συγκεκριμένο βήμα, προσδιορίζουμε τις στατιστικές ιδιότητες (μέση τιμή και διασπορά) όχι μόνο ενός pixel, αλλά και για τα 256 χαρακτηριστικά (εικόνες διάστασης 16 x 16) των ψηφίων του training set με ένδειξη 0. Οι ιδιότητες αυτές θα χρησιμοποιηθούν στη συνέχεια κατά την κατηγοριοποιήση ψηφίων με τον Ταξινομητή Ευκλείδιας Απόστασης.

7. Απεικόνιση Ψηφίου 0 με χρήση της Μέσης Τιμής

Απεικονίζουμε το ψηφίο 0 με χαρακτηριστικά τις μέσες τιμές των χαρακτηριστικών για όλα τα ψηφία 0 που εμφανίζονται στο training set. (Σχήμα 3a)

8. Απεικόνιση Ψηφίου 0 με χρήση της Διασποράς

Ομοίως, απεικονίζουμε το ψηφίο 0 με χαρακτηριστικά τις διασπορές των χαρακτηριστικών για όλα τα ψηφία 0 που εμφανίζονται στο training set. (Σχήμα 3b) Παρατηρούμε ότι και στις δύο περιπτώσεις το ψηφίο ’0’ είναι ευδιάκριτο. Κάνοντας χρήση της μέσης τιμής, η τελική εικόνα δίνει μια **blurred εκδοχή** του ψηφίου, γεγονός αναμενόμενο καθώς προκύπτει από τη μέση τιμή όλων των pixels για τα ψηφία 0.



(a) (b)

Figure 3: Απεικόνιση Ψηφίου 0 με χρήση (a) της Μέσης Τιμής (b) της Διασποράς.

Από την άλλη, κάνοντας χρήση της διασποράς, λαμβάνουμε μια εικόνα με **έντονα τα περιγράμματα - σύνορα** του ψηφίου ’0’, καθώς σε αυτά τα pixels η διασπορά τιμών είναι μεγαλύτερη.

9. Απεικόνιση όλων των Ψηφίων με χρήση της Μέσης Τιμής

Ακολούθως, αφου προσδιορίσουμε τη μέση τιμή και τη διασπορά των 256 χαρακτηριστικών για κάθε μια από τις 10 κλάσεις στο training set, απεικονίζουμε ένα τυχαίο δείγμα για κάθε ψηφίο 0-9 με χρήση της μέσης τιμής των features του. (Σχήμα 4)

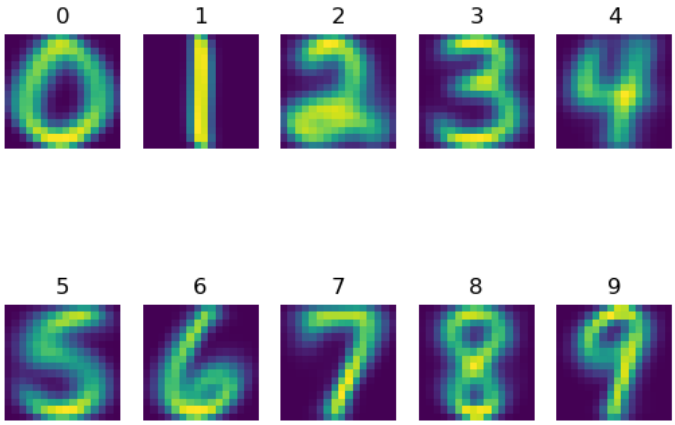


Figure 4: Τυχαία Δείγματα των Ψηφίων 0-9 με χρήση της Μέσης Τιμής των features.

.

10. Ταξινόμηση Ψηφίου βάσει Ευκλείδιας Απόστασης

Στη συνέχεια, καλούμαστε να ταξινομήσουμε το υπ' αριθμόν 101 ψηφίο των test δεδομένων σε μία από τις 10 κατηγορίες (κάθε ένα από τα 10 ψηφία, 0-9, αντιπροσωπεύει μία κατηγορία) βάσει της Ευκλείδειας απόστασης.
Ο **Ευκλείδειος ταξινομητής** χρησιμοποιεί τους μέσους όρους κάθε κλάσης (class means) και για κάθε δείγμα υπολογίζει τις ευκλείδειες αποστάσεις από όλα τα class means ως:

$$D(s,m) = \sqrt{(s_1 - m_1)^2 + ...(s_i - m_i)^2 + ...(s_n - m_n)^2} \quad (1)$$

όπου:
s_i: Το i-οστό feature του δείγματος
m_i: Το i-οστό feature του class mean

Στη συνέχεια, ταξινομεί τα δείγματα στην κλάση από της οποίας το μέσο όρο απείχε λιγότερο (minimum Euclidean distance from class means). Στην προκειμένη περίπτωση, τόσο τα δείγματα όσο και οι μέσοι όροι των κλάσεων αποτελούνται από 256 χαρακτηριστικά.
Επιχειρώντας να ταξινομήσουμε το υπ' αριθμόν 101 ψηφίο των test δεδομένων σε μία από τις 10 κατηγορίες με χρήση της Ευκλείδιας Απόστασης λαμβάνουμε:

Gold Label of Test Digit 101: **6**
Predicted Label of Test Digit 101: **0**

Όπως φαίνεται, λοιπόν, ο Ευκλείδιος Ταξινομητής **αποτυγχάνει** να ταξινομήσει ορθώς το εν λόγω ψηφίο. Αν απεικονίσουμε το ψηφίο αυτό (Σχήμα 5) βλέπουμε ότι ακόμα και ένας άνθρωπος θα δυσκολευόταν να κατηγοριοποιήσει το εν λόγω ψηφίο, γεγονός που δικαιολογεί την αστοχία του ταξινομητή.

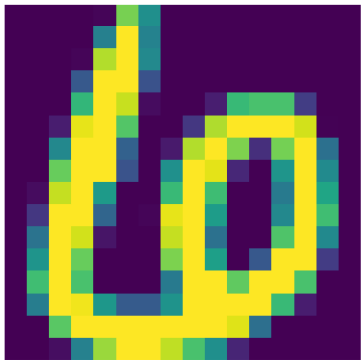


Figure 5: Υπ' αριθμόν 101 ψηφίο των test δεδομένων.

.

11. Ταξινόμηση του Test Dataset βάσει Ευκλείδιας Απόστασης

Ομοίως με πριν, ταξινομούμε όλα τα δείγματα του test dataset στις δέκα κλάσεις του προβλήματος με χρήση της Ευκλείδιας Απόστασης. Το ποσοστό ευστοχίας του Ευκλείδιου Ταξινομητή προκύπτει ίσο με **81.42%**, το οποίο είναι αρκετά ικανοποιητικό, αφήνοντας ωστόσο περιθώρια βελτίωσης.

12. Ευκλείδιος Ταξινομητής ως scikit-learn Estimator

Στο συγκεκριμένο βήμα, υλοποιούμε τον Ταξινομητή Ευκλείδιας Απόστασης ως κλάση της κατηγορίας scikit-learn Estimators που υλοποιεί τις εξής μεθόδους:

• **fit(self, X, y)**: μέθοδος που υπολογίζει τα class means του προβλήματος με βάση τις μέσες τιμές των χαρακτηριστικών των δειγμάτων του συνόλου X.
• **predict(self, X)**: μέθοδος που εκτιμά την κλάση κάθε δείγματος του συνόλου X με βάση την Ευκλείδια Απόσταση του από τα class means.
• **score(self, X, y)**: μέθοδος που αξιολογεί την ευστοχία του ταξινομητή πάνω στο σύνολο X με βάση τα gold labels του συνόλου y.

13. Αξιολόγηση του Ευκλείδιου Ταξινομητή

13a. 5-fold cross-validation Score

Προκειμένου να λάβουμε μια πιο αμερόληπτη και λιγότερο αισιόδοξη αξιολόγηση της επίδοσης του ταξινομητή Ευκλείδιας Απόστασης, ακολουθούμε τη μέθοδο του 5-fold cross-validation [1]. Τα βήματα της διαδικασίας είναι τα εξής:

1. Ανακατέβουμε με τυχαίο τρόπο το σύνολο δεδομένων εκπαίδευσης.
2. Χωρίζουμε το σύνολο αυτό σε k ομάδες.
3. Για κάθε ξεχωριστή ομάδα:
   - Χρησιμοποιούμε μια ομάδα ως το test data set.
   - Χρησιμοποιούμε τις υπόλοιπες k-1 ομάδες ως το training data set.
   - Εκπαιδεύουμε το μοντέλο πάνω στο training data set και αξιολογούμε τη επίδοση του στο test data set.
   - Διατηρούμε το evaluation score και πετάμε το μοντέλο.
4. Συνοψίζουμε λαμβάνοντας ως συνολικό score τον μέσο όρο των επιμέρους scores.

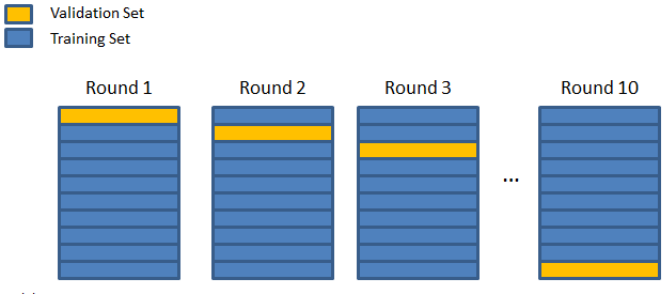


Figure 6: Η τεχνική k-folds Cross Validation για k = 10. [2]

.

Αξιολογώντας, λοιπόν, τον Ταξινομητή Ευκλείδιας Απόστασης με τη μέθοδο του cross-validation για **5 folds** του training data set λαμβάνουμε:

Mean Cross-Validation Accuracy Score: **0.8486** ή **84.86%**
Cross-Validation Error: **0.1514** ± **0.0018**

Παρατηρούμε ότι πετυχαίνουμε **ελαφρώς αυξημένο ποσοστό ευστοχίας** σε σχέση με τη μέθοδο train-test split, καθώς τόσο η εκπαίδευση όσο και η αξιολόγηση του ταξινομητή γίνεται χρησιμοποιώντας partitions του συνόλου `X_train`, το οποίο περιλαμβάνει πολύ περισσότερα δείγματα από το `X_test`.

13b. Περιοχές Απόφασης Ευκλείδιου Ταξινομητή

Σε επόμενο στάδιο, καλούμαστε να σχεδιάσουμε τις περιοχές απόφασης για τις 10 κλάσεις του Ευκλείδιου ταξινομητή. Δεδομένου ότι κάθε δείγμα εκφράζεται από 256 χαρακτηρισιτκά, αρχικά εφαρμόζουμε την **unsupervised τεχνική μείωσης διαστατικότητας PCA** [3], προκειμένου να μεταβούμε σε δείγματα 2 διαστάσεων.
Απεικονίζοντας τις περιοχές απόφασης του ταξινομητή (Σχήμα 7), παρατηρούμε ότι τα σύνορα κάθε κλάσης είναι κάπως δυσδιάκριτα, δεδομένου ότι πραγματευόμαστε ένα multiclass classification πρόβλημα με 10 κλάσεις.

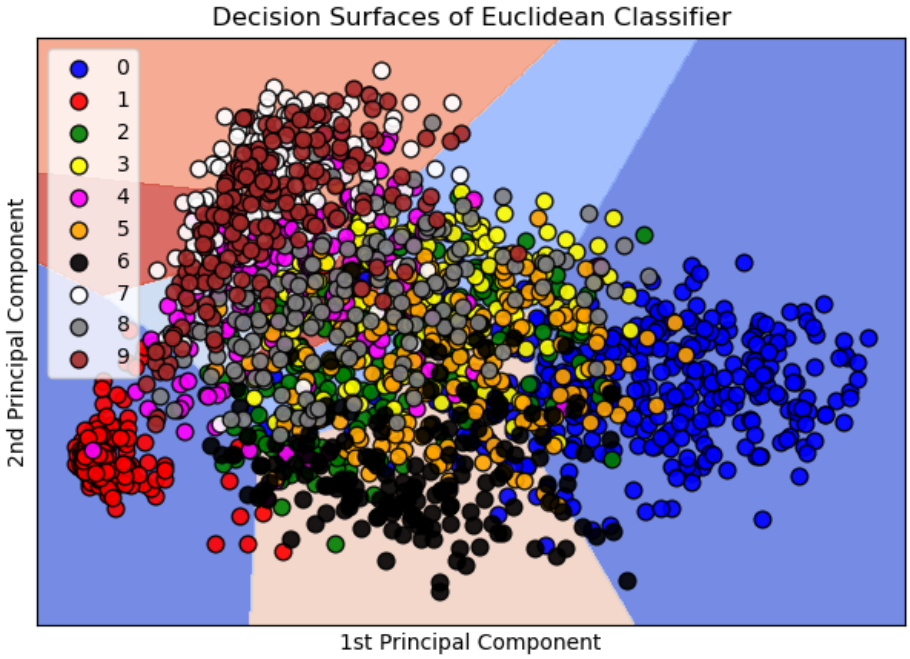


Figure 7: Περιοχές Απόφασης Ευκλείδιου Ταξινομητή.

.

13γ. Καμπύλη Εκμάθησης Ευκλείδιου Ταξινομητή

Τέλος, σχεδιάζουμε την καμπύλη εκμάθησης του Εκλείδιου Ταξινομητή (Σχήμα 8) πάνω στο σύνολο `X_train`, για 5 διαφορετικά υποσύνολά του. Παρατηρούμε ότι καθώς αυξάνονται τα διαθέσιμα δείγματα του training data set, η τυπική απόκλιση του accuracy score μειώνεται, ενώ το χάσμα μεταξύ Training score και Cross-validation score γεφυρώνεται και **συγκλίνει στην τιμή 85%**. Η τροφοδότηση, λοιπόν, του ταξινομητή με δείγματα εκπαίδευσης πέραν των 6000 φαίνεται να μην έχει νόημα, καθώς το ποσοστό ευστοχίας φαίνεται να σταθεροποιείται.

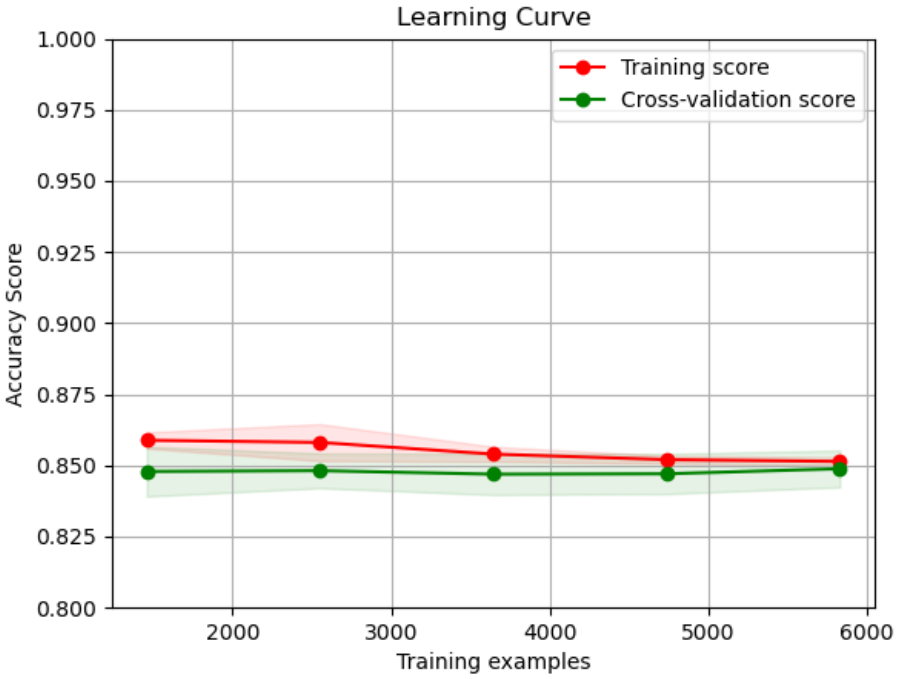


Figure 8: Καμπύλη Εκμάθησης Ευκλείδιου Ταξινομητή.

.

Βήματα Κυρίως Μέρους

14. Προσδιορισμός a-priori πιθανοτήτων

Ως πρώτο στάδιο στην υλοποίηση ενός Naive Bayes ταξινομητή, υπολογίζονται οι a-priori πιθανότητες για κάθε κατηγορία (class priors) πάνω στο σύνολο δεδομένων X_train:

$$Pr(C = i) = \frac{\#\text{samples in class } C_i}{\#\text{samples in dataset}}, \qquad i = 0, \ldots, 9 \tag{2}$$

Ειδικότερα, βρίσκουμε:

Pr(C = 0) = 0.1637, P(C=1) = 0.1378, P(C=2) = 0.1002, P(C=3) = 0.0902, P(C=4) = 0.0894

Pr(C = 5) = 0.0762, P(C=6) = 0.0911, P(C=7) = 0.0885, P(C=8) = 0.0743, P(C=9) = 0.0883

Παρατηρούμε ότι το συχνότερο ψηφίο είναι το 0, ενώ εκείνο που εμφανίζεται πιο σπάνια είναι το 8.

15. Υλοποίηση Naive Bayesian Classifier

Ακολούθως, χρησιμοποιώντας τις τιμές της μέσης τιμής και διασποράς, όπως υπολογίστηκαν στο Βήμα 9 της προπαρασκευής, υλοποιούμε έναν **Naive Bayesian ταξινομητή** με στόχο την κατηγοριοποίηση των ψηφίων του X_test στις 10 κλάσεις του προβλήματος.
Ένας Bayesian ταξινομητής στηρίζεται στον κανόνα απόφασης μέγιστης a-posteriori πιθανότητας (**Maximum a-posteriori Probability - MAP Decision Rule**) με βάση τον Κανόνα του Bayes:

$$Pr(C_i|\mathbf{X}) = \frac{Pr(\mathbf{X}|C_i)Pr(C_i)}{Pr(\mathbf{X})} \tag{3}$$

όπου:
- $Pr(C_i|\mathbf{X})$: a-posteriori Probability
- $Pr(\mathbf{X}|C_i)$: Likelihood Probability
- $Pr(C_i)$: a-priori Probability of class C_i
- $Pr(\mathbf{X})$: Evidence Probability of Feature Vector

Δεδομένου ότι η μεγιστοποίηση του άνωθι κλάσματος γίνεται με βάση την επιλεγμένη κλάση, ο παρονομαστής δεν παρουσιάζει κάποιο ενδιαφέρον στην τελική πρόβλεψη του εκτιμητή. Ως εκ τούτου, ο κανόνας απόφασης γίνεται:

$$y = \underset{i}{\operatorname{argmax}}\, Pr(C_i|\mathbf{X}) = \underset{i}{\operatorname{argmax}}\, Pr(\mathbf{X}|C_i)Pr(C_i) \tag{4}$$

Ο Naive Bayes Classifier τώρα, υποθέτει πως τα επιμέρους "στοιχεία" κάθε δείγματος είναι ανεξάρτητα. Αυτό πέραν της μαθηματικής απλούστευσης στην περιγραφή, καταλήγει και σε μικρότερο αριθμό παραμέτρων για να περιγράψει την κάθε κλάση (με πιθανό κόστος στην απόδοση του ταξινομητή). Δηλαδή ένας Naive Bayes Classifier υποθέτει ότι τα pixels μιας εικόνας είναι **ασυσχέτιστα** και συνεπώς το κάθε ένα μπορεί να περιγραφεί από μια και μόνο πιθανοτική κατανομή [4]. Δεχόμενοι ότι η κατανομή αυτή είναι η **Κανονική**, τα N τω πλήθος features κάθε ψηφίου είναι τυχαίες μεταβλητές **ανεξάρτητες** και **ίδιας κατανομής** (independent & identically distributed) και ο κανόνας απόφασης γίνεται:

$$y = \underset{i}{\operatorname{argmax}}\, Pr(C_i|\mathbf{X}) = \underset{i}{\operatorname{argmax}}\, Pr(C_i) \prod_{j=1}^{N} Pr(X_j|C_i) \tag{5}$$

όπου:

$$Pr(X_j|C_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp -\frac{(X_j - \mu_i)^2}{2\sigma_i^2} \sim N(\mu_i, \sigma_i) \tag{6}$$

Τέλος, δεδομένου ότι η διασπορά για ορισμένα pixels των 10 ψηφίων ισούται με μηδέν, εφαρμόζουμε την τεχνική του **variance smoothing**, προσθέτουμε δηλαδή μια μικρή σταθερά (εδώ $\mathbf{10^{-5}}$) σε κάθε pixel της διασποράς, για λόγους υπολογιστικής ευστάθειας.

Με βάση τα παραπάνω, υλοποιούμε τον NB Classifier συμβατό με τη βιβλιοθήκη scikit-learn και ταξινομούμε όλα τα ψηφία των test δεδομένων:

Custom Gaussian NB Classifier Test Score: **0.7474**

Συγκρίνοντας με την έτοιμη υλοποίηση GaussianNB() της scikit-learn:

scikit-learn Gaussian NB Classifier Test Score: **0.7195**

παρατηρούμε ότι η δική μας υλοποίηση πετυχαίνει **λίγο καλύτερα αποτελέσματα**, γεγονός που κατά πάσα πιθανότητα οφείλεται στην επιλογή καλύτερης τιμής της παραμέτρου variance smoothing (επιλέξαμε $\mathbf{10^{-5}}$ έναντι του default $\mathbf{10^{-9}}$ [5]).

16. Naive Bayesian Classifier με Μοναδιαίο Πίνακα Συνδιακύμανσης

Στο συγκεκριμένο βήμα, ταξινομούμε εκ νέου τα δείγματα του test dataset με χρήση ενός Naive Bayesian ταξινομητή, υποθέτοντας τώρα ότι η διασπορά για όλα τα χαρακτηριστικά, για όλες τις κατηγορίες ισούται με 1. Με άλλα λόγια, ο πίνακας συνδιακύμανσης για τα χαρακτηριστικά κάθε ψηφίου είναι ο **μοναδιαίος πίνακας** ($\Sigma = I$). Δεδομένου ότι κάθε χαρακτηριστικό ακολουθεί την Κανονική Κατανομή, ο Naive Bayesian ταξινομητής ανάγεται σε **Ταξινομητή Ευκλείδιας Απόστασης** και τα decision surfaces είναι **hyperplanes 1ης τάξης** [4]. Κάτι τέτοιο μπορεί να φανεί και από το ποσοστό ευστοχίας του ταξινομητή:

Custom Gaussian NB Classifier Test Score (Unit Variance): **0.8127**

το οποίο είναι πολύ κοντινό στο αντίστοιχο ποσοστό του Βήματος 11 της προπαρασκευής. Παρατηρούμε, μάλιστα, ότι ο εν λόγω ταξινομητής **υπερτερεί** έναντι του απλού Naive Bayesian Classifier που δεν χρησιμοποιεί μοναδιαίο πίνακα συνδιακύμανσης.

17. Σύγκριση Επίδοσης Ταξινομητών

Ακολούθως, συγκρίνουμε την επίδοση των ταξινομητών Naive Bayes, Nearest Neighbors, SVM (με διαφορετικούς kernels) και Ευκλείδιας Απόστασης. Για την αξιολόγηση των ταξινομητών χρησιμοποιούμε την τεχνική cross-validation πάνω στο X_train. Οι επιδόσεις των ταξινομητών συνοψίζονται στον Πίνακα 1.

Classifier	Accuracy (%)
Custom Gaussian Naive Bayes	77.95
Custom Gaussian Naive Bayes (Unit Variance)	84.83
scikit-learn Gaussian Naive Bayes	74.83
Euclidean Distance	84.86
SVM (Linear Kernel)	95.25
SVM (rbf Kernel)	**97.63**
KNN (3 Neighbors)	96.39
KNN (5 Neighbors)	95.99

Table 1: Σύγκριση ευστοχίας ταξινομητών.

Όπως φαίνεται, ο ταξινομητής που πετυχαίνει το καλύτερο ποσοστό ευστοχίας (**97.63%**) στην ταξινόμηση ψηφίων είναι ο **SVM με rbf πυρήνα**, με τον SVM γραμμικού πυρήνα να έπεται. Εξίσου καλά αποτελέσματα επιτυγχάνονται και με χρήση ταξινομητών Πλησιέστερου Γείτονα.

18. Τεχνικές Ensembling

Η βασική ιδέα του βήματος αυτού είναι ο συνδυασμός κάποιων ταξινομητών με αρκετά καλή επίδοση με στόχο να επιτευχθεί επίδοση υψηλότερη των επιμέρους επιδόσεων. Αυτή η τεχνική είναι γνωστή ως **ensembling**. Καθώς είναι σημαντικό οι ταξινομητές που θα συνδυαστούν να χαρακτηρίζονται από διαφορετικό τύπο λαθών, ως πρώτο στάδιο βρίσκουμε τα ψηφία που γίνονται mispredict τις περισσότερες φορές ανά ταξινομητή.

Classifier	Top Mispredictions (left-to-right)
Custom Gaussian Naive Bayes	4 5 3 0 2 8 9 7 6 1
Custom Gaussian Naive Bayes (Unit Variance)	0 2 4 8 5 9 3 7 6 1
scikit-learn Gaussian Naive Bayes	4 3 5 0 2 8 9 6 7 1
Euclidean Distance	0 2 4 8 5 9 3 7 6 1
SVM (Linear Kernel)	5 8 4 3 2 7 6 1 0 9
SVM (rbf Kernel)	3 2 4 8 6 1 7 5 9 0
KNN (3 Neighbors)	4 5 8 2 3 9 7 6 1 0
KNN (5 Neighbors)	4 5 2 8 3 9 7 6 1 0

Table 2: Περισσότερο Mispredicted Ψηφία ανά Ταξινομητή

18α. Ensembling με Voting Classifier

Η πρώτη τεχνική ensembling χρησιμοποιεί έναν μετά-ταξινομητή, ο οποίος συνδυάζει τις επιδόσεις επιμέρους ταξινομητών μέσω ψηφοφορίας για το τελικό αποτέλεσμα (**Voting Classifier**). Διακρίνουμε δύο είδη ψηφοφορίας:

- **Hard Voting**: επιλογή τελικής κλάσης ως αυτήν με τις περισσότερες ψήφους από τους επιμέρους ταξινομητές.
- **Soft Voting**: επιλογή τελικής κλάσης ως αυτήν με το μεγαλύτερο άθροισμα πιθανότητας από τους επιμέρους ταξινομητές.

Σημειώνεται, επίσης, ότι ο αριθμός των ταξινομητών που θα συμμετέχουν στη διαδικασία ψηφοφορίας τύπου **hard voting** πρέπει να είναι **περιττός**, ώστε να αποφεύγονται σενάρια ισοπαλίας. Σε ψηφοφορίες τύπου soft voting μπορεί να γίνει χρήση άρτιου αριθμού ταξινομητών λόγω της χρήσης πιθανοτήτων [6].

Ως πρώτο πείραμα, χρησιμοποιούμε την εξής τριάδα ταξινομητών: **SVM (linear Kernel), KNN (3 neighbours), KNN (5 neighbours)**. Η παραπάνω επιλογή κρίνεται κάπως αφελής, καθώς οι επιμέρους ταξινομητές προβλέπουν εσφαλμένα κατά κανόνα τα ίδια ψηφία (4, 5 και 8), δυσχεραίνοντας έτσι την επίτευξη υψηλότερης επίδοσης. Σημειώνονται τα ακόλουθα ποσοστά ευστοχίας με χρήση cross-validation:

Hard Voting Accuracy: **96.43%**
Soft Voting Accuracy: **96.93%**

Ως δεύτερο πείραμα, χρησιμοποιείται ένας καλύτερος συνδυασμός εκτιμητών που κάνουν συχνότερα mispredict όσο το δυνατόν πιο διαφορετικά ψηφία: **SVM (linear Kernel), SVM (rbf Kernel), KNN (3 neighbours)**. Σημειώνονται τα ακόλουθα ποσοστά ευστοχίας με χρήση cross-validation:

Hard Voting Accuracy: **97.71%**
Soft Voting Accuracy: **97.64%**

Παρατηρεί κανείς ότι και στα δύο πειράματα και οι δύο μέθοδοι ψηφοφορίας καταφέρνουν να επιτύχουν ποσοστά ευστοχίας **ελαφρώς μεγαλύτερα** από το αντίστοιχο ποσοστό του καλύτερου επιμέρους ταξινομητή (96.39% και 97.63% αντιστοίχως). Σημειώνεται, ωστόσο, ότι λόγω της στοχαστικής φύσης του αλγόριθμου αξιολόγησης cross-validation τα παραπάνω ποσοστά μπορεί να παρεκκλίνουν ελάχιστα.

18β. Ensembling με Bagging Classifier

Ως δεύτερη τεχνική ensembling, χρησιμοποιείται ένας **Bagging Classifier**, η λειτουργία του οποίου συνοψίζεται ως εξής: το training dataset χωρίζεται σε τυχαία, πιθανώς επικαλυπτόμενα υποσύνολα και ένας ταξινομητής βάσης εφαρμόζεται σε καθένα από αυτά. Η τελική απόφαση βγαίνει μέσω ψηφοφορίας ή μέσου όρου των προβλέψεων των επιμέρους ταξινομητών.
Ως ταξινομητή βάσης χρησιμοποιούμε αυτόν που έδωσε το καλύτερο ποσοστό ευστοχίας (97.63%), δηλαδή τον **SVM (rbf kernel)**. Αξιολογούμε την επίδοση της εν λόγω τεχνικής με cross validation, συναρτήσει του πλήθους των υποσυνόλων:

5 Estimators Accuracy: **97.60%**
10 Estimators Accuracy: **97.71%**
15 Estimators Accuracy: **97.63%**

Όπως φαίνεται, μόνο στην περίπτωση των 10 υποσυνόλων η τεχνική bagging καταφέρνει να ξεπεράσει ελάχιστα το ποσοστό ευστοχίας του ταξινομητή βάσης.

18γ. Συμπεράσματα - Παρατηρήσεις

Συνοψίζοντας, και οι δύο τεχνικές ensembling δύνανται να βελτιώσουν ελαφρώς το ποσοστό ευστοχίας στο πρόβλημα κατηγοριοποίησης ψηφίων. Ωστόσο, η βελτίωση αυτή είναι αφενός **πολύ μικρή** και αφετέρου **τυχαιοκρατική**, με αποτέλεσμα να προτιμάται η απευθείας χρήση των επιμέρους ταξινομητών καλύτερης επίδοσης, δίχως την χρονική και υπολογιστική πολυπλοκότητα που επιφέρουν οι προαναφερθείσες τεχνικές.

19. Ταξινόμηση με Νευρωνικά Δίκτυα

Στο τελευταίο μέρος, γίνεται χρήση Νευρωνικών Δικτύων για το πρόβλημα ταξινόμησης ψηφίων στις 10 κλάσεις. Έχοντας δημιουργήσει έναν dataloader για να αναλάβει την ανάγνωση των δεδομένων και τον χωρισμό σε batches, υλοποιούμε Fully Connected Νευρωνικά Δίκτυα, πειραματιζόμενοι με τον αριθμό των hidden layers, τον αριθμό νευρώνων κάθε layer και της χρησιμοποιούμενης μη γραμμικής συνάρτησης ενεργοποίησης.
Σε κάθε περίπτωση, γίνεται χρήση ενός **SGD optimizer** με ρυθμό μάθησης **learning rate = 0.1** και **Cross Entropy συνάρτηση κόστους**. Το σύνολο δεδομένων X_train χωρίζεται σε train και validation set, με το πρώτο να χρησιμεύει στην εκπαίδευση του Νευρωνικού με τεχνικές back-propagation και το δεύτερο στη ρύθμιση των υπερπαραμέτρων του.

Για την εκπαίδευση των Νευρωνικών Δικτύων πειραματιζόμαστε με τις μη γραμμικές activation functions **sigmoid(x)**, **tanh(x)** και **ReLU(x)**. Για κάθε μία από αυτές, δημιουργούμε δίκτυα με **2 hidden layers**, το καθένα από **50 ή 100 νευρώνες**, και με **1 hidden layer**, αποτελούμενο από **50 ή 100 νευρώνες**. Στο Σχήμα 9 απεικονίζονται οι συναρτήσεις κόστους για το training & validation set συναρτήσει των 50 εποχών εκπαίδευσης, ενώ στον πίνακα 3 συνοψίζονται τα ποσοστά ευστοχίας κάθε Νευρωνικού Δικτύου στην ταξινόμηση του συνόλου δεδομένων X_test.

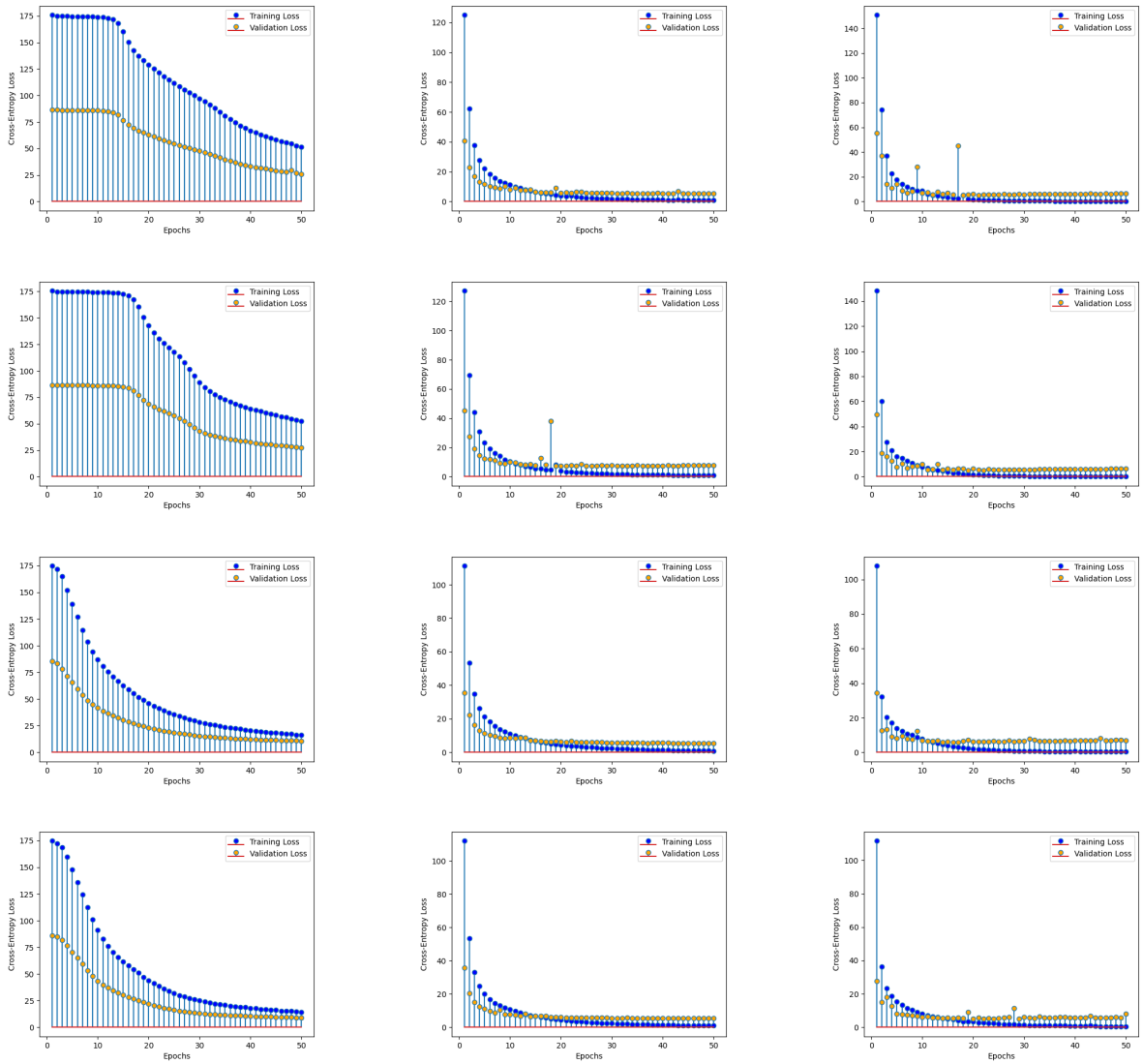


Figure 9: Εξέλιξη του Training & Validation Loss συναρτήσει των εποχών εκπαίδευσης, για συνάρτηση ενεργοποίησης (α) τη sigmoid(x) (πρώτη στήλη), (β) την tanh(x) (δεύτερη στήλη) και (γ) τη ReLU(x) (τρίτη στήλη).

Activation Function	Number of Hidden Layers	Number of Neurons per Layer	Test Accuracy(%)
Sigmoid	2	100	75.386%
tanh	2	100	92.177%
ReLU	2	100	**93.323%**
Sigmoid	2	50	77.180%
tanh	2	50	92.327%
ReLU	2	50	92.875%
Sigmoid	1	100	89.188%
tanh	1	100	92.327%
ReLU	1	100	92.925%
Sigmoid	1	50	90.433%
tanh	1	50	91.779%
ReLU	1	50	90.633%

Table 3: Περισσότερο Mispredicted Ψηφία ανά Ταξινομητή

Με βάση τα παραπάνω δεδομένα προκύπτουν τα εξής συμπεράσματα:

- Η συνάρτηση ενεργοποίησης **sigmoid** φαίνεται να είναι αυτή με τα χαμηλότερα ποσοστά ευστοχίας και τον πιο αργό ρυθμό μάθησης, ενώ είναι η μοναδική εκ των τριων που δίνει καλύτερα αποτελέσματα για λιγότερους νευρώνες και κρυφά επίπεδα, καθώς δεν προλαβαίνει να συγκλίνει επαρκώς σε διάστημα 50 εποχών.
- Η συνάρτηση ενεργοποίησης **tanh** δίνει αρκετά καλύτερα αποτελέσματα (της τάξεως του 90%) ενώ επιτυγχάνει πολύ πιο γρήγορο ρυθμό μάθησης για το Νευρωνικό Δίκτυο.
- Η συνάρτηση ενεργοποίησης **ReLU** φαίνεται να είναι η καλύτερη εκ των τριών επιλογών, καθώς επιτυγχάνει την πιο γρήγορη σύγκλιση, ενώ χρησιμοποιώντας περισσότερους νευρώνες στα κρυφά επίπεδα δίνει το καλύτερο συγκριτικά ποσοστό ευστοχίας (**93.323%**).
- Γενικά, αυξάνοντας τον αριθμό των κρυφών επιπέδων του δικτύου, καθώς και των αριθμό νευρώνων αυτών, η διαδικασία εκπαίδευσης του Νευρωνικού **επιβραδύνεται**, ωστόσο επιτυγχάνονται **καλύτερα ποσοστά ευστοχίας** μόλις επέλθει σύγκλιση.

References

[1] "A gentle introduction to k-fold cross-validation." [Online]. Available: https://machinelearningmastery.com/k-fold-cross-validation/

[2] "Cross validation explained: Evaluating estimator performance." [Online]. Available: https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85

[3] "Principal component analysis." [Online]. Available: https://en.wikipedia.org/wiki/Principal_component_analysis

[4] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[5] "scikit-learn gaussiannb." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB

[6] "How voting classifiers work!" [Online]. Available: https://towardsdatascience.com/how-voting-classifiers-work-f1c8e41d30ff