Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Αναγνώριση Προτύπων: **1η Εργαστηριακή Άσκηση - Οπτική Αναγνώριση Ψηφίων**

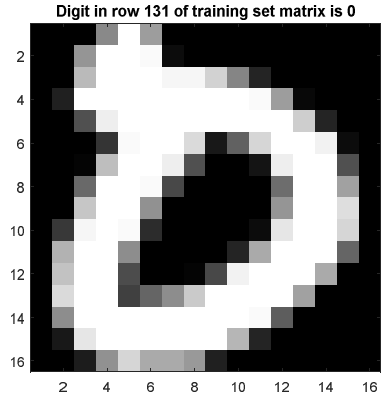
Βελεντζάς Γεώργιος ΑΜ 03105644

Εισαγωγή

Κατα τη διάρκεια της εργαστηριακής άσκησης μελετήθηκαν διαφορετικοί ταξινομητές για την αναγώριση ψηφίων. Χρησιμοποιήθηκαν ψηφία απο την βάση δεδομένων US postal office με 7291 παραδείγματα εκπαίδευσης και 2007 ψηφία για test. Για κάθε ψηφίο δόθηκαν 256 χαρακτηριστικά. Αρχικά προχωρήσαμε σε υλοποίηση και σύγκριση διαφορετικών ταξινομητών, μεταξύ των οποίων ο ευκλείδιος ταξινομητής, ο Bayesian ταξινομητής, ο ταξινομητής k-Nearest Neighbors με ευκελείδια και Mahalanobis απόσταση και με χρήση a-priori πιθανοτήτων, LDA, SVM με πυρήνα γραμμικό και πολυωνυνικό, καθώς και υβριδικοί ταξινομητές, οι οποίοι προκύπτουν απο τον συνδιασμό των παραπάνω. Επίσης δοκιμάσαμε ανάλυση σε πρωτεύουσες συνιστώσες (PCA) ωστε να χρησιμοποιηθούν τα κυρίαρχα χαρακτηριστικά. Παρατηρήθηκε η επίδραση στο συγκεκριμένο πρόβλημα ταξινόμησης και έπειτα προχωρήσαμε σε συνδιασμό των ταξινομητών για την δημιουργία ενος νεου υβριδικού ταξινομητη. Τέλος χρησιμοποιήσαμε τα confidence scores τριών ταξινομητών ως features κάνοντας ενός νέου τύπου dimensionality reduction ωστε να επανεκπαιδεύσουμε το σύστημα με τα νέα δεδομένα. Η καλύτερη απόδοση βρέθηκε 95.32%.

Προπαρασκευή Εργασίας –Συνοπτική παρουσίαση

Κατα τη διαδικασία προπαρασκευής του εργαστηρίου δημιουργήθηκε ένας απλός ευκλείδιος ταξινομητής. Δόθηκαν 7291 δείγματα για εκπαίδευση, με 256 χαρακτηριστικά το καθένα. Τα χαρακτηριστικά αυτά έχουν τιμές στο διάστημα απο -1 έως 1 με την ελάχιστη να δηλώνει μη ύπαρξη μελανιού στην αντίστοιχη περιοχή που χαρακτηρίζει το εικονοστοιχείο. Με τη χρήση του Matlab, μπορούμε να απεικονίσουμε τα δεδομένα για κάθε χαρακτηριστικό σαν μία εικόνα. Όπως φαίνεται στην εικόνα 1 το ψηφίο της γραμμής 131 του πίνακα εκπαίδευσης είναι το ψηφίο «μηδέν». Στη συνέχεια μπορούμε να βρούμε τη μέση τιμή και τη διασπορά για κάθε pixel και για κάθε ψηφίο. Αρχικά δημιουργήσαμε ένα cell, το training_cell το οποίο στο στοιχείο i περιείχε όλα τα training examples που ανήκαν στο ψηφίο i. Συγκεκριμένα το training_cell{i} θα είναι ένας πίνακας Nx256, οπου N το πλήθος των ψηφίων «i-1» στο training set.



Εικόνα 1 : Εμφάνιση του ψηφίου που βρίσκεται στη θέση 131 των δεδομένων εκπαίδευσης

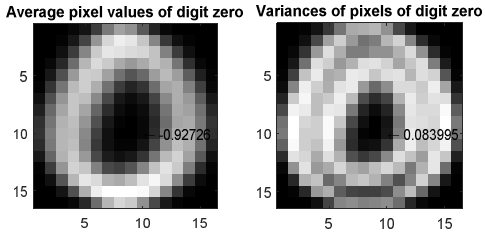
Παρατηρώντας τα αποτελέσματα, η μέση τιμή του pixel(10,10) για το ψηφίο μηδέν έχει την τιμή 0.927 ενώ η μέση τιμή του variance έχει την τιμή 0.0839. Χαμηλή τιμή του variance σημαίνει οτι οι μέσες τιμές είναι αντιπροσωπευτικές, ενω υψηλές τιμές το αντίθετο. Στην εικόνα 2 απεικονίζεται το

ψηφίο «μηδέν» χρησιμοποιώντας τις μέσες τιμές αλλα και τις τιμές της διασποράς. Να σημειώσουμε εδώ οτι η συνάρτηση var(.) του Matlab υπολογίζει την unbiased διασπορά. Έτσι αν με $p_k^d(i,j)$ συμβολίσουμε το pixel στη γραμμή i και στήλη j της εικόνας για το δείγμα k του ψηφίου d, τότε η μέση τιμή και η διασπορά για κάθε pixel θα χαρακτηρίζεται απο τις παρακάτω προφανείς σχέσεις.

$$p_\mu^d(i,j) = \frac{1}{n}\sum_{k=1}^{n} p_k^d(i,j)$$

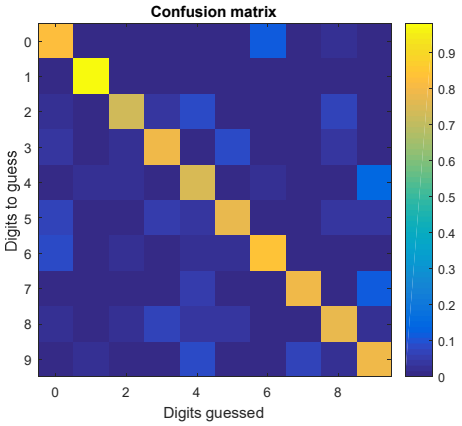
$$p_v^d(i,j) = \frac{1}{n-1}\sum_{k=1}^{n}(p_k^d(i,j) - p_\mu^d(i,j))^2$$

Τα αποτελέσματα φαίνονται στην παρακάτω εικόνα.



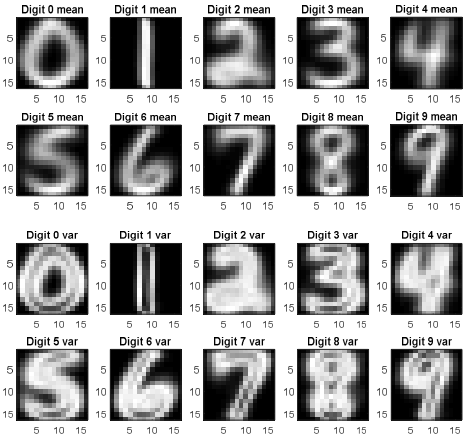
Εικόνα 2: Απεικόνιση του ψηφίου «μηδέν» χρησιμοποιώντας τις μέσες τιμές για κάθε pixel (αριστερά) και τις τιμές τις διασποράς (δεξιά)

Συγκεκριμένα, για κάθε χαρακτηριστικό κάθε ψηφίου βρέθηκε η μέση τιμή, δημιουργώντας έτσι τα «μέσα ψηφία». Στη συνέχεια κάθε ψηφίο για τεστ κατηγοριοποιήθηκε στην κατηγορία με την οποία είχε την μικρότερη ευκλείδια απόσταση. Για την απεικόνιση της απόδοσης του συστήματος αυτού, χρησιμοποιήθηκε ένας confusion matrix 10x10. Στην αρχή ο πίνακας έχει όλα τα στοιχεία του μηδενικά. Κάθε φορά που το ψηφίο «i» κατηγοριοποιείται στο ψηφίο «j» προσθέτουμε «ένα» στο στοιχείο (i,j) του confusion matrix. Στο τέλος κανονικοποιούμε κάθε γραμμή του πίνακα, διαιρώντας τις τιμές της με το σύνολο των αντίστοιχων ψηφίων που τέθηκαν προς κατηγοριοποίηση. Έτσι το στοιχείο (i,j) του πίνακα θα περιέχει το ποσοστό των ψηφίων της γραμμής «i» που κατηγοριοποιήθηκαν στο ψηφίο της γραμμής «j». Ο πίνακας σύγχυσης για τον ευκλείδιο ταξινομητή φαίνεται παρακάτω.



Πίνακας 1: Πίνακας σύγχυσης για τον ευκλείδιο ταξινομητή.

Απο τον πίνακα αυτόν είναι εμφανές ποιά ψηφία είχαν καλή απόδοση και ποιά όχι. Συγκεκριμένα την καλύτερη απόδοση την επιτύχαμε στο ψηφίο «ένα» καθώς τα περισσότερα τεστ ψηφία του «ένα» κατηγοριοποιήθηκαν πράγματι ως «ένα», αλλα επίσης πολύ λίγα άλλα ψηφία κατηγοριοποιήθηκαν ως αυτό (π.χ το «τέσσερα» και το «εννέα»). Κάτι τέτοιο ήταν αναμενόμενο αν παρατηρήσουμε τις εικόνες των «μέσων ψηφίων», αλλα και τις εικόνες των διασπορών του κάθε εικονοστοιχείου για κάθε ψηφίο στην εικόνα 3.

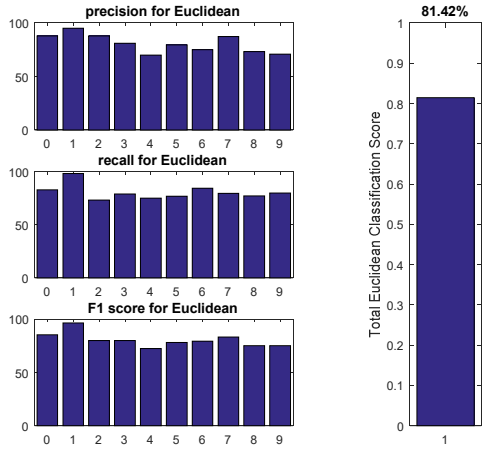


Εικόνα 3 : απεικόνιση των «μέσων ψηφίων» (πάνω) και των διασπορών (κάτω)

Όπως βλέπουμε το ψηφίο «ένα» είχε πολύ ευδιάκριτο μέσο ψηφίο, καθώς και αρκετά μικρές διασπορές εκτός απο τα εικονοστοιχεία στο περίγραμμά του.

Για καλύτερη επισκόπιση των αποτελεσμάτων θα θέλαμε έναν τρόπο να μετρήσουμε την απόδοση του εκάστοτε κατηγοριοποιητή για κάθε ψηφίο ξεχωριστά. Γι'αυτόν τον λόγο θα παρουσιάσουμε τις έννοιες των precision, recall και F1-score. Το recall του ψηφίου «i» ορίζεται ως το ποσοστό των ψηφίων «i» που κατηγοριοποιήθηκαν σωστά, προς τον συνολικό αριθμό των ψηφίων «i» που υπήρχαν στο test set. Για παράδειγμα εαν κάθε ψηφίο το κατηγοριοποιούσαμε στο ψηφίο «i», τότε το recall του ψηφίου «i» θα ήταν 100%. Ωστόσο αυτός δεν θα ήταν καλός κατηγοριοποιητής. Έτσι έχουμε και την παράμετρο του precision. Ως precision του ψηφίου «i» ορίζουμε το ποσοστό των ψηφίων «i» που κατηγοριοποιήθηκαν σωστά, προς τον συνολικό αριθμό των ψηφίων που κατηγοριοποιήθηκαν σαν ψηφίο «i». Υπάρχουν διάφοροι τρόποι να εκτιμησουμε τη συνολική απόδοση ενος κατηγοριοποιητή. Ενας τρόπος είναι να πάρουμε τον συνδιασμό των precision και recall δημιουργώντας μια συνάρτηση score. Μια συνήθης συνάρτηση είναι η F1 score η οποία χαρακτηρίζεται απο τη σχέση 2*precision*recall/(precision+recall). Ο λόγος ύπαρξης του συντελεστή 2 είναι οτι οι τιμές θα είναι κανονικοποιημένες απο το 0 έως το 1. Σε διαφορετική περίπτωση χωρίς την ύπαρξη του συντελεστή η τιμή του F1 θα ήταν απο 0 έως 0.5, στην περίπτωση που και το precision και το recall είχαν την τιμή 1. Στην εικόνα 4 υπάρχουν οι πίνακες precision, recall και F1 score για κάθε ψηφίο, καθώς και η συνολική απόδοση του συστήματος που ορίζουμε ως τα συνολικά ψηφία που κατηγοριοποιήθηκαν σωστά, προς τα συνολικά ψηφία του test set.

Απο την εικόνα αυτή βλέπουμε πως το σύστημα πράγματι πετυχαίνει την καλύτερη απόδοση για το ψηφίο «ένα» ενώ την χειρότερη επίδοση για το ψηφίο «τέσσερα» το οποίο όχι μόνο έχει και πολύ χαμηλό precision, δηλαδή πάρα πολλά ψηφία κατηγοριοποιήθηκαν εσφαλμένα ως 4.



Εικόνα 4: απεικόνιση της απόδοσης με precision (πανω), recall (μέση), F1score (κάτω) και συνολική απόδοση (δεξιά)

Η συνολική απόδοση του συστήματος μετρήθηκε στο 81.42%.

Επέκταση του Ευκλείδιου σε Mahalanobis ταξινομητή

Στη συνέχεια έγινε εκτενέστερη μελέτη στον ευκλείδιο ταξινομητή. Μια αρχική σκέψη είναι να μην γίνει η κατηγοριοποίηση με βάση την ευκλείδια απόσταση, αλλα με βάση την mahalanobis απόσταση των σημείων. Η mahalanobis απόσταση λαμβάνει υπόψιν οτι ο χώρος δεν είναι ομοιόμορφα κατανεμημένος και εξαρτάται απο την διασπορά των χαρακτηριστικών στην εκάστοτε διάσταση.

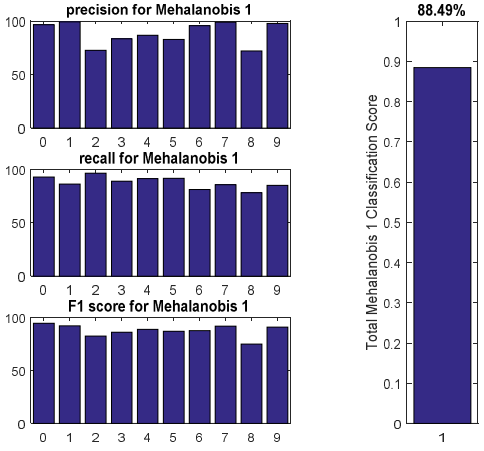
Εδώ πρέπει να κάνουμε την εξής παρατήρηση. Μπορούμε να δοκιμάσουμε δύο περιπτώσεις. Αρχικά η απόσταση του νεου training sample **x** απο την κλάση του ψηφίου «μηδέν» μπορεί να μετρηθεί με δυο διαφορετικούς τρόπους. ο ένας είναι να χρησιμοποιήσουμε τον πίνακα συνδιασποράς των δειγμάτων του ψηφίου «μηδέν» για την έυρεση της mahalanobis απόστασης απο αυτό, και ο δεύτερος να χρησιμοποιήσουμε τον πίνακα συνδιασποράς όλων των δειγμάτων και όλων των ψηφίων στο training set.

Έτσι αν **x** το νέο δείγμα προς ταξινόμηση, $\boldsymbol{\mu_i}$ το μέσο διάνυσμα για το ψηφίο «i», $\boldsymbol{\Sigma_i}$ ο πίνακας συνδιασποράς για τα δείγματα εκπαίδευσης του ψηφίου «i» και **Σ** ο πίνακας συνδιασποράς για όλα τα ψηφία και για όλα τα δείγματα, τότε

$$d_i(x,\mu_i) = \sqrt{(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)}$$

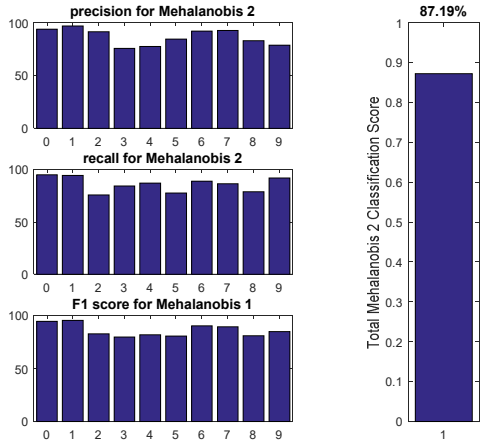
$$d(x,\mu_i) = \sqrt{(x-\mu_i)^T \Sigma^{-1}(x-\mu_i)}$$

είναι δύο διαφορετικοί τρόποι να κανονικο-ποιήσουμε την απόσταση του νέου δείγματος απο τους μέσους. Για λόγους ευκολίας τον πρώτο ταξινομητή θα τον ονομάσουμε mahalanobis-1 ενω τον δεύτερο mahalanobis-2. Πρέπει να επισημάνουμε εδώ πως λόγω της αντιστροφής του πίνακα συνδιασποράς στην κάθε περίπτωση, θέλουμε να αποφύγουμε την περίπτωση να είναι singular κ γι'αυτό προσθέτουμε μια πολύ μικρή σταθερά (10^{-3}) σε όλα τα διαγώνια στοιχεία του. Τα συνολικά αποτελέσματα φαίνονται στις παρακάτω εικόνες.



Εικόνα 5 : απεικόνιση της απόδοσης του Mahalanobis-1 ταξινομητή, με precision(πανω), recall(μέση), F1-score (κάτω) και συνολική απόδοση δεξιά

Η συνολική απόδοση του mahalanobis-1 ταξινομητή φαίνεται στην εικόνα 5 και είναι 88.49% , δηλαδή εμφανώς καλύτερη απο την απόδοση του απλού ευκλείδιου ταξινομητή.



Εικόνα 6: απεικόνισης της απόδοσης του ταξινομητή mahalanobis-2, precision(πάνω), recall(μέση), F1 score(κάτω) και συνολική απόδοση (δεξιά)

Η συνολική απόδοση του mahalanobis-2 ταξινομητή είναι 87.19% και φαίνεται στην εικόνα 6. Και στους δυο κατηγοριοποιητές παρατηρούμε οτι το ψηφίο «τέσσερα» δεν έχει πλέον την χειρότερη απόδοση ενώ το ψηφίο «οχτώ» δεν φαίνεται να βελτιώθηκε. Ο mahalanobis-1 έχει λίγο καλύτερη επίδοση, καθώς λαμβάνει υπόψιν τις διασπορές του κάθε ψηφίου ξεχωριστά. Στο τέλος της εργασίας θα γίνει η εκτίμηση όλων των κατηγοριοποιητών για κάθε ψηφίο ξεχωριστά.

Bayesian classifier

Σε αυτό το μέρος κατασκευάζουμε έναν απλό Bayesian κατηγοριοποιητή δοκιμάζοντας διαφορετικές τεχνικές για την συνδιασπορά των χαρακτηριστικών. Ο Bayesian classifier κατηγοριοποεί το κάθε ψηφίο στην κατηγορία με τη μέγιστη πιθανότητα. Πιο συγκεκριμένα αν **x** το νέο δείγμα προς κατηγοριοποίηση, τότε

$$category = \arg\max_{\omega_i} P(\omega_i \mid x) = \arg\max_{\omega_i} P(x \mid \omega_i) P(\omega_i)$$

οπού οπού ω_i η κατηγορία του ψηφίου i, και

$$P(\mathbf{x} \mid \omega_i) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}_i|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^{\mathrm{T}} \boldsymbol{\Sigma}_i^{-1} (\mathbf{x}-\boldsymbol{\mu}_i)\right)$$

με $\mathbf{\Sigma_i}$ τον πίνακα συνδιασποράς του και $\mathbf{\mu_i}$ τον μέσο της κατηγορίας i. Αρχικά εργαζόμαστε βρίσκοντας τις a-priori πιθανότητες, διαιρώντας για κάθε κατηγορία το πλήθος των δειγμάτων της εκάστοτε κατηγορίας στο training set προς το συνολικό πλήθος των δειγμάτων.

$$P(\omega_i) = \frac{\#\text{samples of class i in training set}}{\#\text{of samples in training set}}$$

Στη συνέχεια με βάση τις υποδείξεις της άσκησης κάνουμε την παραδοχή οτι ο πίνακας συνδιασποράς για κάθε κατηγορία είναι διαγώνιος, με μηδενική συνδιασπορά μεταξύ διαφορετικών χαρακτηριστικών. Με αυτόν τον τρόπο ο υπολογισμός της πιθανοφάνειας γίνεται πιο εύκολος αφου για διαγώνιο πίνακα συνδιασπορών οι γκαουσιανές γίνονται ανεξάρτητες μεταξύ των διαστάσεων όπως φαίνεται στους παρακάτω υπολογισμούς.

$$P(\mathbf{x} \mid \omega_i)P(\omega_i) =$$

$$\frac{1}{(2\pi)^{d/2}\left|\mathbf{\Sigma}_i\right|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^{\mathrm{T}}\mathbf{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right)P(\omega_i) =$$

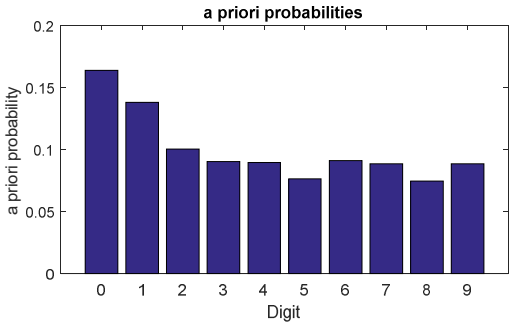
$$P(\omega_i)\prod_{k=1}^{d}\frac{1}{\sqrt{2\pi}\sigma_k^i}\exp\left(-\frac{(x_k-\mu_k^i)^2}{2(\sigma_k^i)^2}\right) \Rightarrow$$

και λογαριθμίζοντας

$$\ln(P(\mathbf{x} \mid \omega_i)P(\omega_i)) =$$

$$\ln(P(\omega_i)) + \sum_{k=1}^{d}\ln\left(\frac{1}{\sqrt{2\pi}\sigma_k^i}\right) - \sum_{k=1}^{d}\frac{(x_k-\mu_k^i)^2}{2(\sigma_k^i)^2}$$

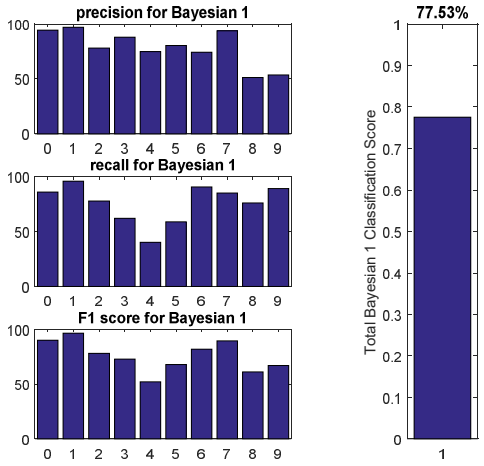
οπου k η διάσταση της συνιστώσας και i το ψηφίο. Με την τελευταία αυτή σχέση τα πράγματα γίνονται υπολογιστικά πιο εύκολα, καθώς σε διαφορετική περίπτωση κατα την υλοποίηση θα έπρεπε να υπολογιστεί η ορίζουσα του πίνακα συνδιασπορών. Ο πίνακας συνδιασπορών όμως είναι διαγώνιος με πολύ μικρά στοιχεία, και η ορίζουσά του που θα είναι το γινόμενο των διαγώνιων αυτών των 256 στοιχείων θα είναι κοντά στο μηδέν. Υπολογιστικά αυτό θα δημιουργούσε πολλά προβλήματα με απειρισμούς κατα τους υπολογισμούς των πιθανοτήτων, και αρκετή απώλεια πληροφορίας απο την απώλεια ακρίβειας δεκαδικών πολύ κάτω απο την υποδιαστολή. Ο λογάριθμος απο την άλλη προσφαίρει πολύ μεγαλύτερη διακριτική ικανότητα σε μικρούς αριθμούς και γι'αυτό προτιμάται σε πολλές εφαρμογές εκτός των άλλων καθώς μετατρέπει το γινόμενο σε άθροισμα. Και πάλι προτιμούμε να προσθέσουμε μια πολύ μικρή σταθερά σε όλα τα διαγώνια στοιχεία του πίνακα, (για να είμαστε δίκαιοι θα προσθέσουμε την ίδια σταθερά αρχικά και ίση με 10^{-3}).



Εικόνα 7: εμφάνιση των a-priori πιθανοτήτων

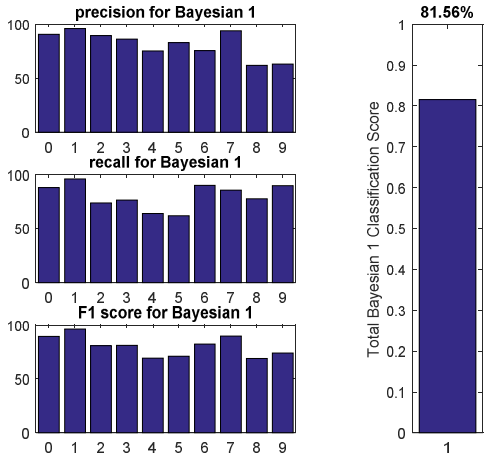
Στην εικόνα 7 εμφανίζονται οι a-priori πιθανότητες απο όπου βλέπουμε οτι είχαμε συνολικά περισσότερα μηδενικά απο οποιοδήποτε άλλο ψηφίο. Έπειτα ακολουθούν οι «άσσοι» και τη μικρότερη a-priori είχε το ψηφίο «οχτώ». Αυτό σημαίνει οτι για το ψηφίο 8 είχαμε λιγότερα δείγματα εκπαίδευσης απο ότι για τα άλλα ψηφια, και αυτός είναι και ένας απο τους λόγους που οι mahalanobis ταξινομητές μας έδειξαν αδυναμία σε αυτό το ψηφίο.

Για λόγους ευκολίας θα ονομάσουμε την περίπτωση που χρησιμοποιήθηκε διαγώνιος πίνακας συνδιασπορών με βάση τις διασπορές των ψηφίων, Bayesian-1, ενώ την περίπτωση που χρησιμοποιήθηκε ο μοναδιαίος πίνακας ώς πίνακας συνδιασπορών Bayesian-2.



Εικόνα 8: απόδοση του Bayesian-1 ταξινομητή, με βάση τα precision, recall, F1-score και συνολικής απόδοσης

Απ'ότι βλέπουμε στην εικόνα 8, στην περίπτωση του Bayesian-1 ταξινομητή η απόδοση είναι μικρότερη απο την απόδοση των άλλων ταξινομητών και ίση με 77.53%. Ο κύριος λόγος είναι η αρχική παραδοχή, ότι τα μη διαγώνια στοιχεία θα είναι μηδενικά. Κάτι τέτοιο δεν ισχύει αν σκεφτούμε την φύση του προβλήματος. Λόγω του πάχους της γραμμής του στυλό ή του μολυβιού, είναι προφανές οτι γειτονικά pixels θα έχουν μη μηδενική συνδιασπορά. Στη συνέχεια δοκιμάσαμε να αυξήσουμε την μεταβλητή που προσθέτουμε στα διαγώνια στοιχεία του πίνακα, προσθέτοντας «τεχνητή» διασπορά.



Εικόνα 9: απόδοση του Bayesian-1 με προσθήκη σταθεράς ίσης με 0.1 στα διαγώνια στοιχεία, αντί για 0.001που είχαμε στις άλλες περιπτώσεις

Κάτι ενδιαφέρον εδω είναι οτι η F1-score απόδοση για κάθε ψηφίο και στις δύο φαίνεται να συσχετίζεται με τις a-priori πιθανότητες της εικόνας 7. Επίσης τα ψηφία «οχτώ» και «εννέα» εμφανίζουν αρκετά κακή απόδοση μαζί με το ψηφίο «τέσσερα». Παρατηρήθηκε οτι αυξάνοντας την τάξη μεγέθους της σταθεράς αυτής, η απόδοση αυξανόταν συνεχώς, μέχρι ένα σημείο. Για σταθερά ίση με 0.1 τα αποτελέσματα ήταν τα καλύτερα δυνατά γι'αυτήν την περίπτωση, με απόδοση ίση με 81.56%.

Στην περίπτωση του Bayesian-2, οπου χρησιμοποιήθηκε μοναδιαίος πίνακας συνδιασπορών, περιμένουμε να έχουμε απόδοση περίπου ίση με την απόδοση του ευκλείδιου ταξινομητή. Άν οι a-priori πιθανότητες ήταν ακριβώς ίσες τότε θα είχαμε ακριβώς την ίδια απόδοση με τον ευκλείδιο ταξινομητή. Αυτό γιατί όπως φαίνεται παρακάτω:

$$\arg\max_{\omega_i}\left(P(\mathbf{x}\mid\omega_i)P(\omega_i)\right)=$$

$$\arg\max_{\omega_i}\left(\frac{1}{(2\pi)^{d/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^{\mathrm{T}}(\mathbf{x}-\boldsymbol{\mu}_i)\right)P(\omega_i)\right)=$$

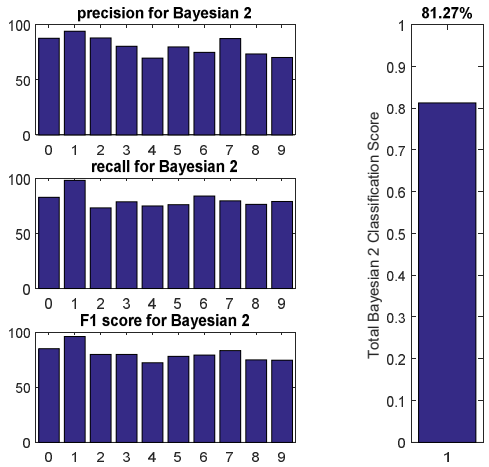
$$\arg\max_{\omega_i}\left[\log\left(\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^{\mathrm{T}}(\mathbf{x}-\boldsymbol{\mu}_i)\right)P(\omega_i)\right)\right]=$$

$$\arg\max_{\omega_i}\left[\log(P(\omega_i))-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^{\mathrm{T}}(\mathbf{x}-\boldsymbol{\mu}_i)\right]=$$

$$\arg\min_{\omega_i}\left[\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^{\mathrm{T}}(\mathbf{x}-\boldsymbol{\mu}_i)-\log(P(\omega_i))\right]$$

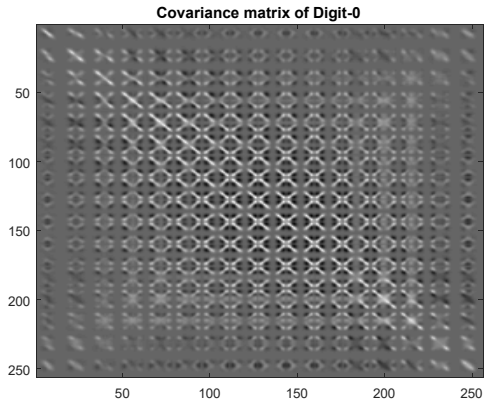
και προφανώς με ίσες a-priori ο όρος μπορεί να απαλειφεί απο την παραπάνω εξίσωση και να προκύψει ισοδύναμη συνθήκη με τον ευκλείδιο ταξινομητή. Ωστόσο η συνολική απόδοση του ταξινομητή Bayesian-2 είναι 81.27%, δηλαδή όχι καλύτερη απο τον ευκλείδιο ταξινομητή. Αυτό σημαίνει οτι οι a-priori πιθανότητες που έχουμε χρησιμοποιήσει, εξάγοντάς τις απο το training set δεν ακολουθούν την ίδια κατανομή στο test set. Αυτό μπορεί να συμβαίνει είτε γιατί το training set δεν είναι αντιπροσωπευτικό των a-priori

πιθανοτήτων των ψηφίων, είτε γιατι το test set είναι αρκετά μικρότερο. Τα αποτελέσματα του Bayesian-2 φαίνονται στην εικόνα 10.



Εικόνα 10: εμφάνιση της απόδοσης του Bayesian-2 ταξινομητή με precision, recall, F1-score, και συνολική απόδοση

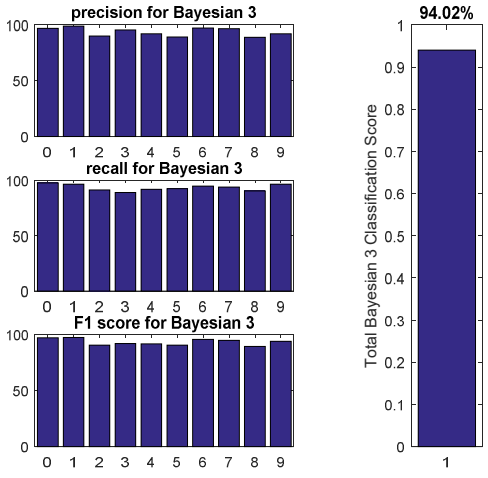
Αρχικά φαίνεται λοιπόν οτι ο Bayesian ταξινομητής δεν υπερτερεί του ευκλείδιου, ή του mahalanobis ταξινομητή που παρουσιάσαμε. Ένα τέτοιο συμπέρασμα όμως θα ήταν εσφαλμένο καθώς όπως είπαμε δεν χρησιμοποιήσαμε τους πλήρεις πίνακες συνδιασπορών. Στην εικόνα 11 εμφανίζουμε τον πλήρη πίνακα συνδιασποράς του ψηφίου «μηδέν».



Εικόνα 11 : πλήρης πίνακας συνδιασποράς του ψηφίου «μηδέν» ως εικόνα. Οι πιο λευκές τιμές χαρακτηρίζουν μεγάλες συνδιασπορές.

Απο την είκόνα 11 παρατηρούμε κάτι αναμενόμενο. Τα γειτονικά εικονοστοιχεία έχουν μεγάλες συνδιασπορές. Αυτό απεικονίζεται με τα χαρακτηριστικά «Χ». Λόγω του τρόπου αποθήκευσης της πληροφορίας, το εικονοστοιχείο 16n ωστόσο δεν είναι γειτονικό με το εικονοστοιχείο 16n+1, και γι'αυτό ο πίνακας έχει αυτή τη δομή.

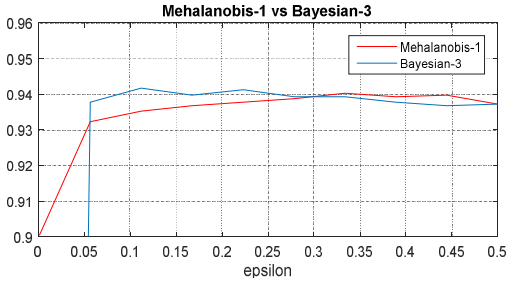
Η παράμετρος των συνδιασπορών δεν λαμβάνεται υπ'οψιν στον Bayesian-1 και στον Bayesian-2. Γι'αυτόν τον λόγο στη συνέχεια υλοποιούμε ξανά τον αλγόριθμο χρησιμοποιώντας τους πλήρεις πίνακες συνδιασπορών. Τον ταξινομητή αυτόν θα τον ονομάσουμε Bayesian-3. Η δυσκολία εδω βρίσκεται στο γεγονός οτι ο υπολογισμός των πιθανοτήτων θα είναι πολύ κακός και θα πρέπει να αυξήσουμε τη σταθερά που προσθέτουμε στα διαγώνια στοιχεία του πίνακα για να αποφύγουμε τα συσσωρευόμενα υπολογιστικά σφάλματα. Με σταθερά 0.1 τα αποτελέσματα είναι πολύ ικανοποιητικά καθώς η απόδοση φτάνει στο 94.02% ενω όπως είδαμε για τον Bayesian-1 η απόδοση ήταν 81.56%. Για λόγους σύγκρισης θα επανέλθουμε και στους mahalanobis ταξινομητές αλλάζοντας τη σταθερά αυτή για να μελετήσουμε την απόδοσή τους.



Εικόνα 12: Απόδοση του Bayesian-3 ταξινομητή με χρήση σταθεράς 0.1

Χρησιμοποιώντας την ίδια σταθερά στον mahalanobis-1 ταξινομητή, προκύπτει απόδοση 93.47%, δηλαδή αρκετά κοντά στην απόδοση του bayesian-3 ταξινομητή.

Μέχρι στιγμής λοιπόν φαίνεται οτι ο Bayesian-2 ταξινομητής και ο mahalanobis-1 επιτυγχάνουν απόδοση περίπου 94% μετά την προσθήκη μιας σταθεράς στα διαγώνια στοιχεία του πίνακα συνδιασπορών. Η εξάρτηση αυτή φαίνεται στα παρακάτω διαγράμματα.

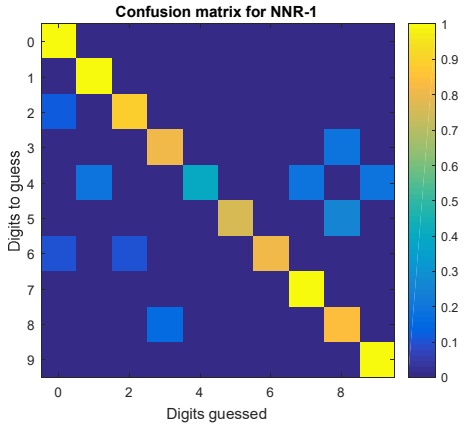


Διάγραμμα 1: Σύγκριση mahalanobis-1 (κόκκινη γραμμή) και Bayesian-3(μπλέ γραμμή) μετά την προσθήκη σταθεράς epsilon στα διαγώνια στοιχεία του πίνακα συνδιασπορών.

Απο το διάγραμμα 1 παρατηρούμε οτι ο Bayesian επιτυγχάνει πολύ καλή απόδοση πάνω απο 94% για σταθερά περίπου 0.11, αλλα η απόδοση είναι μηδενική όταν η σταθερά είναι πολύ μικρή. Αντίθετα ο mahalanobis-1 επιτυγχάνει την καλύτερή του επίδοση για σταθερά περίπου 0.33.

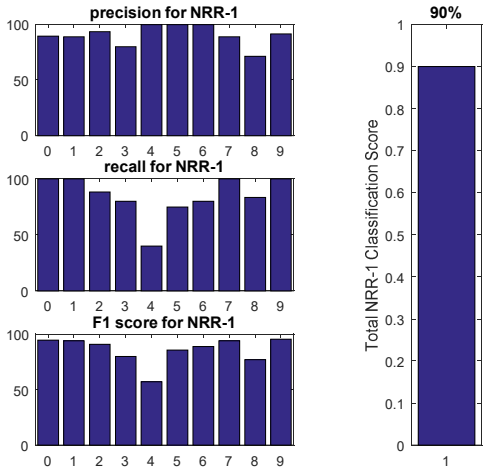
k-NN ταξινομητές, με ευκλείδια απόσταση και με χρήση a-priori πιθανοτήτων.

Σε αυτό το μέρος υλοποιούμε τον αλγόριθμο των Κ κοντινότερων γειτόνων. Για κάθε νέο δείγμα του τεστ σετ, ψάχνουμε τους Κ πιο κοντινούς (με βάση κάποιο μετρικό) γείτονες απο το training set. Το νεο δείγμα θα κατηγοριοποιηθεί στην κατηγορία που έχει τους περισσότερους γείτονες με αυτό. Στην αρχή θα χρησιμοποιήσουμε μονο 1000 training samples, και θα κάνουμε τεστ σε 100 test samples εφαρμόζοντας NNR-1.



Πίνακας 2: πίνακας σύγχυσης για NNR-1 χρησιμοποιώντας μόνο τα 1000 πρώτα δείγματα εκπαίδευσης και τα πρώτα 100 δείγματα για τεστ

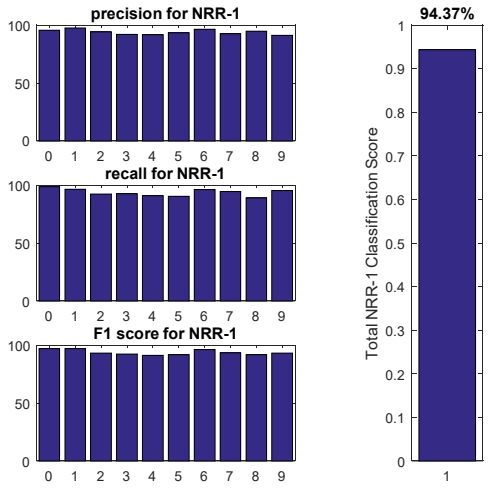
Απο τον πίνακα 2 παρατηρούμε πολύ καλή απόδοση του NNR-1 αν λάβουμε υπ'όψιν την απλότητά του. Και πάλι το ψηφίο 1 φαίνεται να έχει αρκετά καλό precision και recall, αλλα για καλύτερη επισκόπιση των τιμών ας δούμε κάθε γραμμή και κάθε στήλη του πίνακα αυτού ξεχωριστά.



Εικόνα 13: απόδοση του NNR-1 με χρήση 1000 δεδομένων train και 100 δεδομένων τεστ.

Απο τα αποτελέσματα παρατηρούμε πολύ καλή απόδοση της τάξης του 90% για τα περισσότερα ψηφία εκτος απο το «τέσσερα» και το «οχτώ». Το «οχτώ» είχε πολύ λίγα δείγματα στο train set και αυτός είναι ένας λόγος. Το «τέσσερα» απο την άλλη βλέπουμε οτι έχει precision 100% (αυτό το

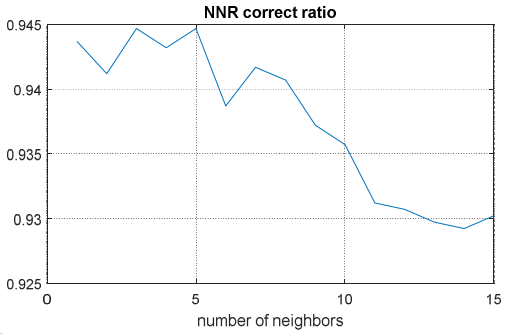
βλέπουμε και απο τον πίνακα σύγχυσης), δηλαδή κανένα απο τα ψηφία που δεν ήταν «τέσσερα» δεν κατηγοριοποιήθηκαν εσφαλμένα ως «τέσσερα». Αντίθετα το recall του είναι πολύ χαμηλό, δηλαδή αρκετά «τεσσάρια» κατηγοριοποιήθηκαν εσφαλμένα σε άλλα ψηφία. Στη συνέχεια επαναλαμβάνουμε την παραπάνω διαδικασία χρησιμοποιώντας όλα τα δείγματα του train set και όλα τα δείγματα του test set.



Εικόνα 14: απόδοση του NNR-1 χρησιμοποιώντας όλα τα δεδομένα test και train

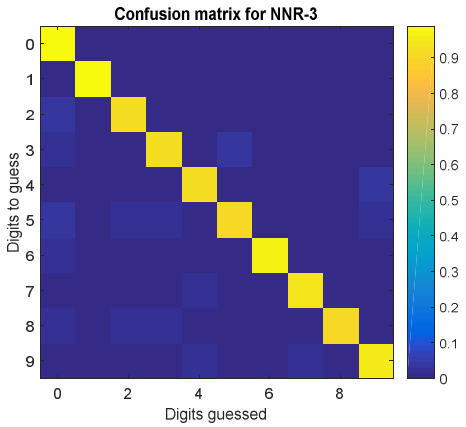
Τελικά χρησιμοποιώντας τον NNR-1 στα πλήρη σύνολα train και test, επιτυγχάνουμε απόδοση 94.37%. Ο kNN είναι αρκετά καλός αλγόριθμος και απλός και συνήθως χρησιμοποιείται. Αρνητικό χαρακτηριστικό του είναι οτι σε εφαρμογές πραγματικού χρόνου θα είναι πολύ αργός. Αυτό γιατί για κάθε νέο δείγμα θα πρέπει να βρίσκει τους κοντινότερους γείτονες, ελέγχοντας όλα τα training samples, ενώ οι παραπάνω αλγόριθμοι χρειάζονται μεν χρόνο για να γίνει η εκπαίδευση, αλλα έπειτα η απόφαση λαμβάνεται με απλούστερους υπολογισμούς, η τάξη μεγέθους των οποίων εξαρτάται απο το πλήθος των κλάσεων και όχι απο το πλήθος των παραδειγμάτων παρατήρησης.

Στη συνέχεια εφαρμόζουμε τον αλγόριθμο για διαφερετικό πλήθος γειτόνων, και ελέγχουμε την απόδοση. Να αναφέρουμε εδώ οτι στις υλοποιήσεις πρέπει να αποφεύγονται τα for loops διαφορετικά ο αλγόριθμος θα είναι εξαιρετικά αργός. Τελικά για διαφορετικές τιμές του k βρέθηκαν οι αποδόσεις και αποθηκεύτηκαν σε ένα array του οποίου τις τιμές παρουσιάζονται παρακάτω.

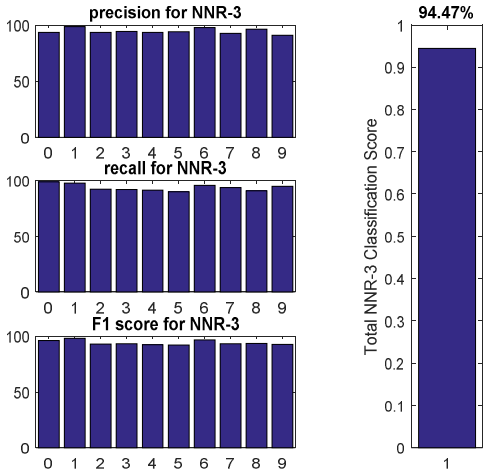


Διάγραμμα 2: εμφάνιση απόδοσης ως συνάρτηση του αριθμού των κοντινότερων γειτόνων

Από τις μετρήσεις παρατηρήσαμε οτι η απόδοση για 3 ή 5 γείτονες ήταν η ίδια και ίση με 94.47%. Στην εικόνα 15 φαίνονται αναλυτικότερα τα precision, recall και F1-score.



Πίνακας 3:πίνακας σύγχυσης για τον αλγόριθμο NNR-3



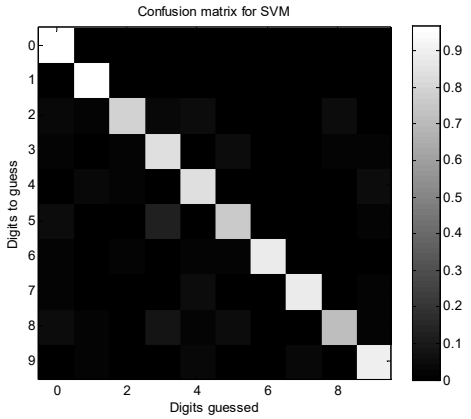
Εικόνα 15: απόδοση του NNR-3 με precision, recall, F1-score και συνολική απόδοση

Στη συνέχεια προτείνουμε ώς βελτιωμένη παραλλαγή του NNR να λάβουμε υπ'όψιν τις a-priori πιθανότητες. Αν για παράδειγμα στην περίπτωση του NNR-7, 4 απο τους γείτονες ήταν «μηδέν» και τρεις ήταν «εννέα», τότε το δείγμα θα το κατηγοριοποιούσαμε στο «μηδέν», όμως η a-priori πιθανότητα του μηδέν είναι σχεδόν διπλάσια απο του 9 οπότε κάτι τέτοιο δεν θα ήταν σωστό. Μία καλή σκέψη είναι να βρίσκουμε την πλειοψηφία διαιρώντας πρώτα το πλήθος των ψήφων κάθε κατηγορίας με την a-priori πιθανότητα της κάθε κατηγορίας. Η απόδοση του NNR-3 με χρήση a-priori πιθανοτήτων μετρήθηκε συνολικά 94.67%, όσο και του NNR-7. Η αύξηση 0.2% δηλώνει 4 συνολικά πιο σωστές ταξινομήσεις. Η συνάρτηση αυτή είναι υλοποιημένη σαν NNRk.m και θα την χρησιμοποιήσουμε στο τέλος με τους υβριδικούς ταξινομητές.

Support Vector Machines

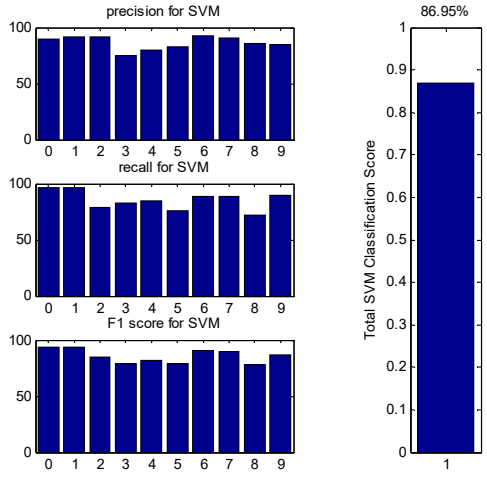
Σε αυτό το στάδιο χρησιμοποιήθηκαν support vector machines μέσω της βιβλιοθήκης bioinformatics toolbox του matlab. Λόγω υψηλών υπολογιστικών απαιτήσεων χρησιμοποιήθηκε η μέθοδος least squares για τον υπολογισμό του εκάστοτε ταξινομητή. Δημιουργήθηκαν συνολικά 10 ταξινομητές σαν στοιχεία ενός cell Svm{.} , ο καθένας απο τους οποίους ελέγχει αν το νέο δείγμα ανήκει στην αντίστοιχη κατηγορία. Για παράδειγμα ο ταξινομητής Svm{1} επιστρέφει «1», άν το δείγμα προς έλεγχο ανήκει στην κατηγορία-1 (δηλαδή του ψηφίου μηδέν). Για κάθε νέο δείγμα γίνεται ταξινόμηση απο όλα τα Svm και στο τέλος λαμβάνεται μία απόφαση. Ωστόσο στα προβλήματα multiclass classification οπού χρησιμοποιείται υλοποίηση του τύπου one-vs-all, χρειαζόμαστε κάποια καλή στρατηγική απόφασης μεταξύ ισοβαθμίας ψήφων ή και καθόλου ψήφων. Συγκεκριμένα υπάρχει η περίπτωση κανένας απο τους ταξινομητές να μην απαντήσουν θετικά για κάποια κατηγορία, δηλαδή όλοι να αποφασίζουν οτι ανήκουν στις υπόλοιπες κατηγορίες. Επίσης υπάρχει περίπτωση πάνω απο δυο ταξινομητές να απαντήσουν θετικά. Σε αυτές τις περιπτώσεις θα μπορούσαμε να απαντήσουμε με βάση τις a-priori πιθανότητες, ωστόσο έγινε λίγο πιο εκτενής παρατήρηση. Τα support vector machines στο matlab χρησιμοποιούν την συνάρτηση svmclassify η οποία επιστρέφει μόνο την απόφαση. Στον κώδικα της svmclassify παρατηρήθηκε οτι χρησιμοποιεί την svmdecision, η οποία υπολογίζει και επιστρέφει κάποια margins, όμως η svmclassify δεν τα χρησιμοποιεί. Πρώτο βήμα λοιπόν ήταν να τροποποιηθεί η svmclassify ωστε να εξάγουμε και τα margins απο το σύνορο διαχωρισμού μεταξύ των κλάσεων. Η δεύτερη τροποποίηση είναι εντός της svmdecision. Η svmdecision επιστρέφει και το score του εκάστοτε svm. Άν το score αυτό είναι αρνητικό (ή μηδέν) ταξινομεί το δείγμα στην κύρια κατηγορία και η έξοδος είναι «1», ενώ αν είναι θετικό αποφασίζει «0». Το score αυτό σχετίζεται με την απόσταση απο το σύνορο απόφασης, και γι'αυτό τον λόγο τροποποιούμε τον κώδικα ωστε να εξάγουμε και αυτό το χαρακτηριστικό στην svmclassify. Στον κώδικα της εργασίας η svmclassifyandscore είναι η τροποποιημένη svmclassify. Η svmdecision είναι επίσης στον φάκελο εργασίας και δεν πρέπει να αφαιρεθεί καθως είναι private function.Για κάθε νέο δείγμα **x** καλούμε την svmclassifyandscore για καθε μηχανη Svm{i} και αποθηκεύουμε τα scores σε ενα array. Τελικά κατηγοριοποιούμε το δείγμα στην κατηγορία που έχει το μικρότερο score. (Το

«μικρότερο» ή «μεγαλύτερο» δεν έχει νόημα εδώ. Είναι προφανές οτι το matlab χρησιμοποιεί συνάρτηση που επιστρέφει τιμές μικρότερες του μηδενός αν το δείγμα ανήκει στην κατηγορία και μεγαλύτερες αντίθετα).



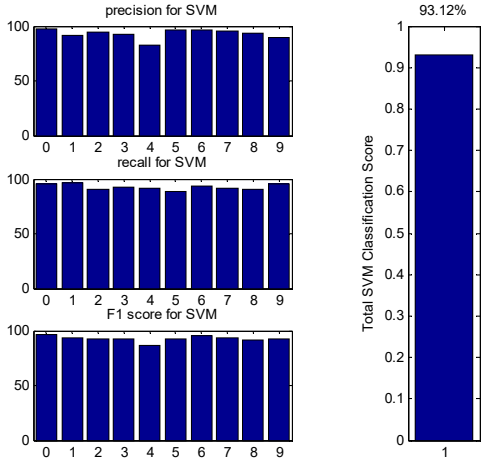
Πίνακας 4: πίνακας σύγχυσης για τα Support Vector Machines με γραμμικό πυρήνα

Απο τον παραπάνω πίνακα σύγχυσης φαίνεται οτι και τα support vector machines με χρήση γραμμικού πυρήνα, επιτυγχάνουν σχετικά καλή κατηγοριοποίηση αλλα όχι καλύτερη απο την Bayesian-3 και την mahalanobis-1 υλοποίηση. Συγκεκριμένα το ποσοστό είναι 86.95% ενώ αναλυτικότερα τα χαρακτηριστικά φαίνονται στην παρακάτω εικόνα.



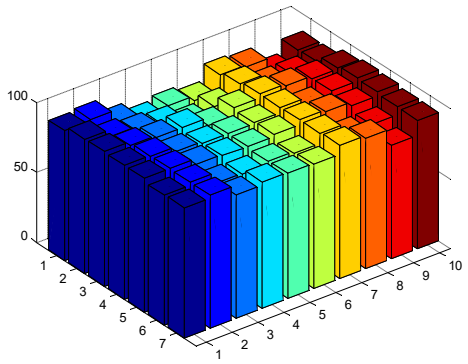
Εικόνα 17: precision, recall, F1-score και συνολική απόδοση των SVM με γραμμικό πυρήνα

Στη συνέχεια έγινε εκπαίδευση και έλεγχος και για πολυωνυμικούς πυρήνες. Τα αποτελέσματα για τρίτου βαθμού πυρήνα φαίνονται παρακάτω.Η απόδοσή του μετρήθηκε αρκετά καλή και ίση με 93.12%



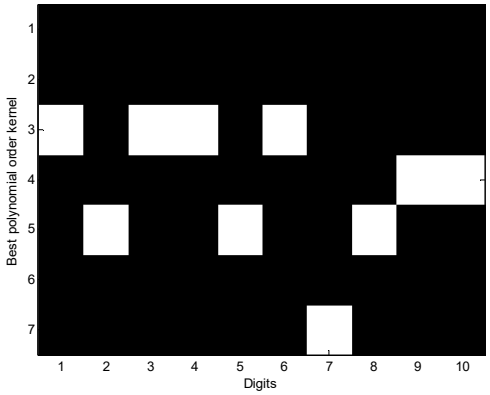
Εικόνα 18: Απόδοση SVM με πολυωνυμικό πυρήνα τρίτου βαθμού

Οι struct δομές για κάθε περίπτωση και κάθε κατηγορία έχουν αποθηκευτεί, ωστε να μην χρειάζεται να επαναλαμβάνουμε τη διαδικασία εκπαίδευσης κάθε φορά. Στον κώδικα το κομμάτι της εκπαίδευσης είναι σχολιασμένο, και αντί αυτού γίνεται load των ταξινομητών κάθε φορά. Παρακάτω φαίνεται η συνολική απόδοση ανάλογα με τον βαθμό του πολυωνύμου του πυρήνα που χρησιμοποιήθηκε. Τα παρακάτω υλοποιούνται στο Svm_evaluations.m.



Εικόνα 19: F1-score για πολυωνυμικούς πυρήνες απο 1 (γραμμικός) εως 7 και για κάθε ψηφίο ξεχωριστά

Οι δύο καλύτεροι απο τους πυρήνες ήταν οι πολυωνυμικοί τρίτου βαθμού και οι πολυωνυμικοί πέμπτου βαθμού. Συγκεκριμένα οι πολυωνυμικοί τρίτου βαθμού είχαν απόδοση 93.12% ενώ οι πολυωνυμικοί πέμπτου βαθμού είχαν απόδοση 93.42%. Σε σχέση με τα F1-scores για το κάθε ψηφίο ξεχωριστά μπορούμε να βρούμε τη θέση του μεγίστου κάθε στήλης του διαγράμματος της εικόνας 19. Τα αποτελέσματα φαίνοντα στην εικόνα 20 και απ'ότι βλέπουμε ο πολυωνυμικός πυρήνας τρίτου βαθμού πετυχαίνει τα 4 καλύτερα F1-scores ενω του πέμπτου βαθμού πετυχαίνει τα τρία καλύτερα.

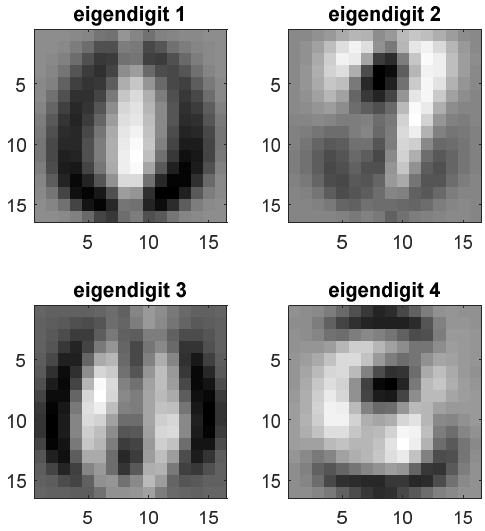


Εικόνα 20: Με λευκό απεικονίζεται ο καλύτερος κατα F1-score πολυωνυμικός πυρήνα; (ο βαθμός του οποίου αντιστοιχεί στη γραμμή του πίνακα) για το εκάστοτε ψηφίο (με το οποίο υπάρχει αντιστοιχία κατα στήλη).

Στη συνέχεια ζητήθηκε να κατασκευαστεί άλλος ένας ταξινομητής. Πριν απο αυτό προτιμήθηκε να προχωρήσουμε σε περεταίρω ανάλυση των χαρακτηριστικών με ανάλυση σε πρωτεύουσες συνιστώσες όπως παρουσιάζεται στο παρακάτω τμήμα της αναφοράς. Στη συνέχεια χρησιμοποιούμε τα νέα δεδομένα για επανεκπαίδευση ωστε να δούμε την πιθανή επίδραση στην απόδοση.

Principal Component Analysis.

Σε αυτό το μέρος προχωρήσαμε σε επεξεργασία των χαρακτηριστικών ωστε να μειωθεί η διάστασή τους, κρατώντας την χρήσιμη πληροφορία, απεικονίζοντας τα χαρακτηριστικά σε έναν χώρο χαμηλότερης διάστασης. Η βάση αυτού του χώρου περιγράφεται απο τα ιδιοδιανύσματα του πίνακα συνδιασποράς όλων των ψηφίων και όλων των δειγμάτων, τα οποία καλούμε «ιδιοψηφία». Το κάθε νέο δείγμα μπορεί να γραφεί ως ένας γραμμικός συνδιασμός των ιδιοψηφίων αυτών. Εαν κρατήσουμε μόνο λίγα απο τα ιδιοψηφία, αλλα τα πιο σημαντικά, μπρούμε να περιγράψουμε με αρκετά καλή ακρίβεια το κάθε δείγμα σε αυτή τη βάση. Τα παρακάτω θα τα βρείτε στο m-file με τίτλο eigendigits.



Εικόνα 21: εμφάνιση των τεσσάρων πρώτων βασικότερων ιδιοψηφίων

Αν $\boldsymbol{\Sigma}$ είναι ο πίνακας συνδιασποράς των δειγμάτων και $\mathbf{e_i}$ το ιδιοδιάνυσμα i, $\boldsymbol{\lambda_i}$ η ιδιοτιμή του αντίστοιχου ιδιοδιανύσματος, τότε θα κρατήσουμε τα k πιο σημαντικά ιδιοδιανύσματα. Σημαντικό θεωρείται το ιδιοδιάνυσμα που έχει μεγάλη ιδιοτιμή. Αρχικά υπολογίζουμε τη συνολική μέση τιμή για όλα τα δείγματα και όλα τα χαρακτηριστικά $\mathbf{x_m}$. Τα νέα μετασχηματισμένα

train και test δεδομένα θα έχουν k χαρακτηριστικά με

$$y_i = \mathbf{e}_i^T (\mathbf{x} - \mathbf{x}_m)$$

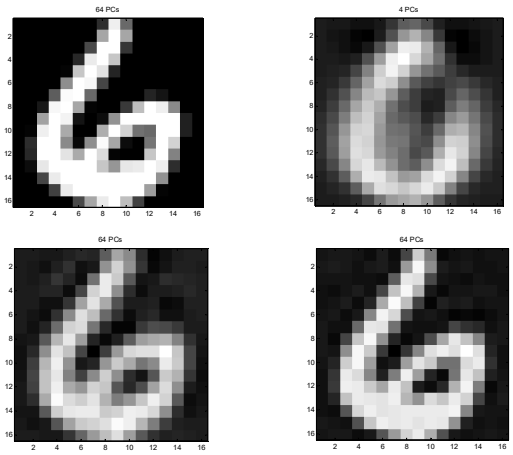
να είναι το i-οστό χαρακτηριστικό του δείγματος **x**. Έτσι κατασκευάζουμε έναν πίνακα **E** με τα κυριότερα k ιδιοδιανύσματα στις στήλες του (δίνεται η δυνατότητα στον κώδικα να αλλάξετε αυτή τη διάσταση). Κάθε δείγμα εκπαίδευσης και κάθε δείγμα για τεστ μετασχηματίζεται μέσω της παρακάτω σχέσης

$$\mathbf{y} = \mathbf{E}^T (\mathbf{x} - \mathbf{x}_m)$$

Είναι ενδιαφέρον εδω να δείξουμε οτι πράγματι ένα μικρός αριθμός αριθμός απο πρωτεύουσες συνιστώσες είναι αρκετός για να περιγράψει το κάθε δείγμα. Μπορούμε να επιστρέψουμε στον αρχικό χώρο πολλαπλάσιάζοντας απο αριστερά την σχέση με τον πίνακα ιδιοδιανυσμάτων και στη συνέχεια προσθέτοντας τον μέσο

$$\hat{\mathbf{x}} = \mathbf{E}\mathbf{y} + \mathbf{x}_m$$

Παρακάτω φαίνεται η επανακατασκευή ενός τυχαίου ψηφίου μετά απο αυτή τη διαδικασία, κρατώντας 4, 64 και 128 πρωτεύοντα χαρακτηριστικά.



Εικόνα 22: αρχική εικόνα(πάνω αριστερά), ανακατασκευασμένη εικόνα απο 4 συνιστώσες (πάνω δεξιά), ανακατασκευασμένη εικόνα απο 64 συνιστώσες (κάτω αριστερά), και απο 128 συνιστώσες (κάτω δεξιά).

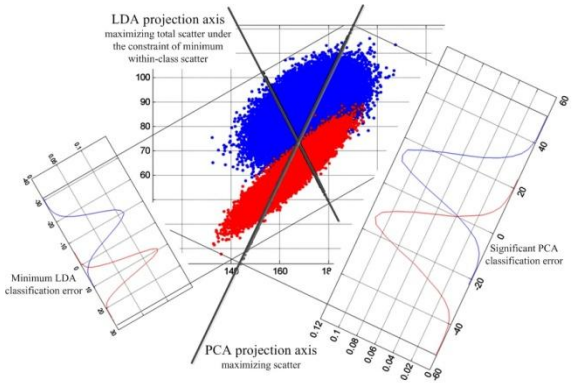
Στη συνέχεια εκπαιδεύσαμε svm με πολυωνυμικό πυρήνα πρώτου και τρίτου βαθμού χρησιμοποιώντας τις 64 πρώτες συνιστώσες, αλλα τα αποτελέσματα ήταν χειρότερα απ'ότι με τα πλήρη δεδομένα, με 88.1% και 86.2% απόδοση αντίστοιχα.

Η κακή επίδραση της pca στα support vector machines μας οδηγεί στο να αφήσουμε την παραπάνω μελέτη καθώς στα πλαίσια μιας εργαστηριακής άσκησης θα ήταν εξαιρετικά χρονοβόρα.

LDA classification

Ο επόμενος ταξινομητής που θα δοκιμάσουμε είναι linear discriminant analysis. Σε αυτή τη μέθοδο η λογική είναι σχεδόν παρόμοια με την

pca, μόνο που εδώ ψάχνουμε τις διαστάσεις στις οποίες τα δεδομένα είναι περισσότερο διαχωρίσιμα. Αυτή η διάσταση δεν έιναι πάντα η διάσταση με την μεγαλύτερη διασπορά, ούτε συσχετίζεται με αυτή.

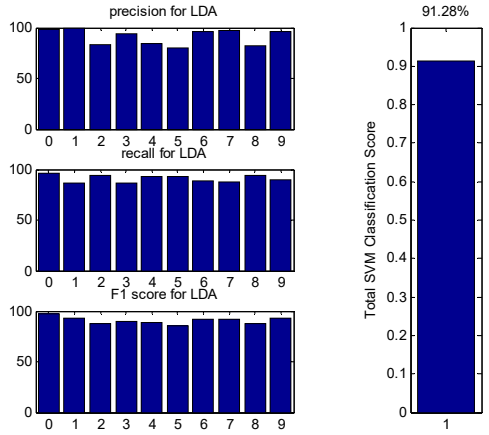


Εικόνα 23: pca vs lda . Στην lda η άξονες προβολής των δεδομένων είναι οι βέλτιστοι ωστε οι κλάσεις να είναι πιο διαχωρίσιμες (source : www.ait.edu.gr)

Για τον ταξινομητή αυτόν δοκιμάστηκαν διαφορετικές μέθοδοι και τελικά οι καλύτερες επιδόσεις παρατηρήθηκαν όταν χρησιμοποιήθηκε η επιλογή mahalanobis απόστασης. Το πρόβλημα που υπήρχε εδώ είναι οτι οι πίνακες συνδιασποράς δεν βρίσκονταν θετικά ορισμένοι.

Πριν την χρήση lda προχωρήσαμε σε μείωση διαστάσεων με ανάλυση σε πρωτεύουσες συνιστώσες. Παρατηρήθηκαν πολυ μεγάλα ποσοστά επιτυχίας ακόμα και για πολύ λίγες συνιστώσες. Τελικά για 22 πρωτεύουσες συνιστώσες η συνολική απόδοση μετρήθηκε 91.28%.

Τελικά η LDA σε συνδιασμό με PCA είναι μια αρκετά καλή μέθοδος και αρκετά γρήγορη. Ωστόσο χρειάζεται περισσότερη μελέτη και αρκετό χρόνο προσομοίωσης ωστε να υπολογιστούν οι βέλτιστοι παράμετροι. Οι βέλτιστοι παράμετροι εκπαίδευσης θα πρέπει να υπολογίζονται απο cross validation και όχι απευθείας με έλεγχο της απόδοσης στο test set.



Εικόνα 24: απόδοση της lda μετά απο μείωση διάστασης σε 22 χαρακτηριστικά με pca.

Συνολική εκτίμηση και σχολιασμός ταξινομητών.

Τελικά παρατηρούμε οτι οι ταξινομητές με τη συνολική καλύτερη απόδοση είναι ο NNR-3 με mahalanobis, ο Bayesian-3 με smoothing στα διαγώνια στοιχεία, τα SVM με πολυωνυμικό πυρήνα 3^{ου} βαθμού, η LDA μετά απο μείωση χαρακτηριστικών σε 22 με PCA, και επίσης και ο NNR-1 που θεωρείται πολύ καλός λόγω ταχύτητας. Συνολικά οι επιδόσεις απο τον έλεγχο πάνω στα test δεδομένα φαίνονται στον παρακάτω πίνακα.

NNR-3 (or 7) + a-priori	94.67%
NNR-1 euclidean	94.47%
Bayesian smoothed	94.02%
Mahalanobis-1	93.47%
SVM – Poly3	93.12%
LDA with PCA-22	91.28%
NNR-1 mahalanobis	90.00%

Πίνακας 5: συνολικές αποδόσεις των καλύτερων ταξινομητών απο αυτούς που ελέχθηκαν.

Υβριδικοί ταξινομητές

Στη συνέχεια υλοποιούμε εκ νέου τρεις απο τους ταξινομητές ως συναρτήσεις, οι οποίες θα δέχονται ως ορίσματα το training set, μαζί με το group στο οποίο ανήκει το κάθε δείγμα, καθώς και ένα δείγμα απο το test set, και θα επιστρέφει τις πιθανότητες να ανήκει αυτό το δείγμα στην κάθε κατηγορία. Έτσι θα είναι πιο εύκολο να διαχειριστούμε την έξοδο του κάθε ταξινομητή και να αποφασίσουμε για την κατηγοριοποίηση του νέου δείγματος. Για confidence scores θα χρησιμοποιήσουμε τα F1-scores όπως την ορίσαμε στο πρώτο μέρος της εργασίας. Ώς ταξινομητές θα χρησιμοποιήσουμε τον NNR-7 με a-priori πιθανότητες, τον smoothed Bayesian και τα SVM. Ο λόγος είναι η διαφορετική τους φύση και επίτευξη διαφορετικών scores για κάθε κατηγορία μεταξύ τους.

Πλειοψηφική απόφαση ταξινομητών

Για την υλοποίηση αυτής της μεθόδου (βλέπε HybridClassifier.m) δημιουργούμε έναν πίνακα out ο οποίος είναι διάστασης 3x2007. Κάθε γραμμή αντιστοιχεί σε κάποιον ταξινομητή (η πρώτη στον NNR-7 με a-priori, η δεύτερη στον Bayesian και η τρίτη στα SVM) και κάθε στήλη σε ένα test sample. Το στοιχείο out(i,j) για παράδειγμα είναι η απόφαση του ταξινομητή i για το test sample j. Με αυτόν τον τρόπο μπορούμε εύκολα να αποφασίσουμε για κάθε test sample. Στην περίπτωση που δεν υπάρχει πλειοψηφική απόφαση μπορούμε να δοκιμάσουμε διάφορες στρατηγικές. Η μία είναι να λάβουμε στην τύχη μια απόφαση, η άλλη είναι να λάβουμε υπ'όψιν

την a_priori πιθανότητα αποφασίζοντας υπέρ του ψηφίου με την μέγιστη, ή υπέρ του ψηφίου με την ελάχιστη a_priori. Επίσης θα μπορούσαμε να αποφασίζουμε υπέρ του ταξινομητή που είχε το καλύτερο συνολικό σκορ, ή πάντα υπέρ κάποιου ταξινομητή γενικώς. Συνολικά τα αποτελέσματα φαίνονται στον παρακάτω πίνακα.

max a-priori	94.87%
min a-priori	94.97%
NNR-7 ours	**95.42%**
Bayesian	94.62%
SVM	94.67%

Πίνακας 6: Εμφάνιση συνολικού ποσοστού Voting Classifier, για διαφορετικές στρατηγικές σε περίπτωση ισοβαθμίας ψήφων

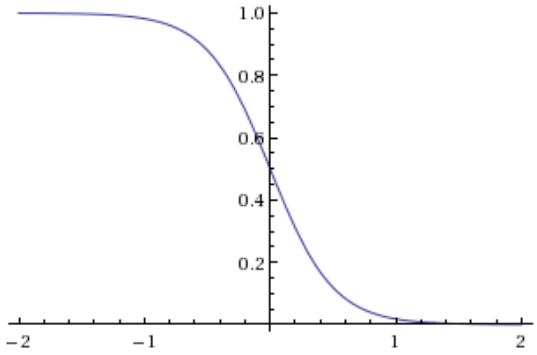
Στον πίνακα 6 φαίνεται οτι την καλύτερη επίδοση την πετύχαμε όταν σε περίπτωση ισοβαθμίας επιλέξαμε την ταξινόμηση του τροποποιημένου NNR-7, οποίος είχε και συνολικά την καλύτερη επίδοση όπως είδαμε απο τα προηγούμενα μέρη.

Συνδιασμένοι ταξινομητές βεβαιότητας.

Στη συνέχεια θα πρέπει να εξάγουμε κάποιου τύπου confidence απο τον κάθε ταξινομητή. Στην περίπτωση του Bayesian θα αντιστοιχήσουμε το confidence με την a-porsteriori πιθανότητα. Λόγω του οτι στους υπολογισμούς μας δεν εξάγουμε την πιθανότητα αλλα τον λογάριθμο αυτής, θα επιλέξουμε να κανονικοποιήσουμε κάθε φορά τα αποτελέσματα στο πεδίο απο 0.05 έως 0.95, που συσχετίζονται με το σφάλμα και την απόδοση του Bayesian. Ήδη βέβαια με αυτόν τον τρόπο η στρατηγική max θα μας επιστρέφει σχεδόν πάντα τον Bayesian ταξινομητή. Γι'αυτόν τον λόγο θα μειώσουμε το άνω όριο σε 0.8, το οποίο όπως θα δούμε θα δώσει αρκετά καλά αποτελέσματα. Στην περίπτωση του NNR-7 θα ξεκινήσουμε προσθέτωντας μια πολύ μικρή βεβαιότητα σε κάθε ψηφίο, και ίση με 0.05. Ίση δηλαδή περίπου με το συνολικό σφάλμα του NNR-7. Στη συνέχεια για κάθε γείτονα θα προσθέτουμε 0.90/7, δηλαδή 0.13 στην αντίστοιχη κλάση. Έτσι για παράδειγμα αν έχουμε και τους 7 γείτονες στην κλάση 0, θα έχουμε confidence 0.13x7+0.05=0.96 ενώ σε κάθε άλλη κλάση θα έχουμε 0.05. Οι τιμές αυτές δεν είναι αναγκαίο να αθροίζουν στη μονάδα καθώς μιλάμε για βεβαιότητα και όχι πιθανότητα. Στην περίπτωση των SVM όπως είχαμε αναφέρει μπορούμε να εξάγουμε το margin απο το boundary. Συγκεκριμένα όταν το margin αυτό είναι αρνητικός αριθμός το δείγμα ανήκει στην εν λόγω κλάση ενώ όταν είναι θετικός δεν ανήκει. Υπάρχουν διαφορετικές μελέτες για το πως θα εξαχθούν οι a-posteriori απο αυτές τις τιμές. Στην συγκεκριμένη περίπτωση θα μετασχηματίσουμε τους αριθμούς αυτούς στο (0,1) μέσω της σιγμοειδούς συνάρτησης

$$f(x) = -\frac{1}{1+e^{-ax}} + 1$$

οπου a είναι μια σταθερά που καθορίζει πόσο απότομη είναι αυτή η σιγμοειδής. Έτσι αν η έξοδος απο το SVM είναι πολύ αρνητικός αριθμός, τότε η σιγμοειδής θα μας επιστρέψει αριθμό κοντά στη μονάδα, ενώ είναι θετικός θα μας επιστρέψει αριθμό κοντά στο μηδέν.



Εικόνα 25: Σιγμοειδής συνάρτηση για μετασχηματισμό των margins του SVM σε βεβαιότητες, με σταθερά a=4.

Στη συνέχεια λοιπόν χωρίζουμε το training set σε δύο σύνολα, το train και το validation set. Επιλέγουμε ως validation set τα πρώτα 1450 παραδείγματα και ως training set τα υπόλοιπα.

Απο το ιστόγραμμα του training set παρατηρούμε οτι έχουμε επιλέξει αρκετά καλό δείγμα εκπαίδευσης το οποίο μοιάζει με την κατανομή των a-priori πιθανοτήτων. Στη συνέχεια εκπαιδεύουμε τον κάθε ταξινομητή. Μάλιστα πριν προβούμε σε γραμμικό συνδιασμό ταξινομητών. Ελέγχουμε τον κάθε ταξινομητή ξεχωριστά με βάση το validation set και αποθηκεύουμε τον πίνακα σύγχυσης κάθε ταξινομητή σαν πίνακα ωστε να τον χρησιμοποιήσουμε αργότερα σε διαφορετική υλοποίηση. Με εκπαίδευση πάνω στο νεο training set και έλεγχο στο validation set οι ταξινομητές NNR-7, Bayesian και SVM-poly3 είχαν απόδοση 96.76%, 96.97% και 96.55% αντίστοιχα.

Κατα τη φαση απόφασης για κάθε ψηφίο, αποθηκεύουμε τα confidence scores για κάθε ταξινομητή. Συγκεκριμένα έχουμε δημιουργήσει το cell conf, οπου το conf{i}(:,j) έχει τα confidence scores του ταξινομητή i για την απόφαση του δείγματος j. Για παράδειγμα για το 6ο κατα σειρά δείγμα έχουμε τα παρακάτω confidence scores για κάθε ταξινομητή.

	NNR7	Bayesian	SVM
0	**0.4571**	**0.8000**	**0.0664**
1	0.0500	0.0500	0.0607
2	0.0500	0.6970	0.0177
3	0.0500	0.6669	0.0273
4	0.0500	0.6612	0.0246
5	0.0500	0.6831	0.0179
6	**0.5929**	**0.7870**	**0.6261**
7	0.0500	0.5158	0.0085
8	0.0500	0.7282	0.0239
9	0.0500	0.5800	0.0329

Πίνακας 7 : confidence scores για το 6ο ψηφίου του validation set για κάθε ταξινομητή

Οι δύο πρώτοι ταξινομητές θα αποφάσιζαν οτι το ψηφίο είναι «μηδέν» ενώ ο SVM θα αποφάσιζε οτι είναι «έξι». Όμως οι δύο πρώτοι έχουν μεγάλα confidences και για τα δύο ψηφία, ενώ ο SVM έχει χαμηλά confidences για το ψηφίο «μηδέν». Τελικά το ψηφίο είναι το ψηφίο «έξι» και ο SVM θα το ταξινομούσε σωστά. Προφανώς η αξιοπιστία της ταξινόμησης συσχετίζεται με το πόσο «κοντά» είναι το confidence του δεύτερου ψηφίου. Στη συνέχεια ας δούμε τι αποτελέσματα θα πάρουμε για διαφορετικούς συνδιασμούς των confidences.

Θα δούμε τα αποτελέσματα των max, min,mean και γινομένου. Με max εννοούμε οτι για κάθε ταξινομητή ελέγχουμε το μέγιστο confidence, συγκρίνουμε τα μέγιστα confidence των ταξινομητών και απαντάμε με βάση των ταξινομητή που έχει το μέγιστο απο τα μέγιστα confidences (δηλαδη max max). Με min απαντάμε με βάση τον ταξινομητή που έχει το ελάχιστο μέγιστο confidence (max min) . Με mean υπολογίζουμε τον μέσο όρο τον confidences των ταξινομητών και απαντάμε με το μέγιστο, και με max υπολογίζουμε πρώτα τα γινόμενα. Με έλεγχο πάνω στο validation set τα αποτελέσματα φαίνονται στον παρακάτω πίνακα.

max	96.28%
min	96.07%
mean	97.52%
product	97.52%
diff	96.90%

Πίνακας 8: απόδοση του υβριδικού ταξινομητή στο validation set για διαφορετικές συναρτήσεις

Στη συνέχεια, και με βάση την παρατήρηση του πίνακα 7, θα αποφασίσαμε να λαμβάνουμε υπ'όψιν τον ταξινομητή που έχει το μεγαλύτερο κενό μεταξύ του μέγιστου confidence και του δεύτερου μέγιστου confidence. Η τελική απόδοση πάνω στο validation set μετρήθηκε ίση με 96.90% όπως φαίνεται στον πίνακα 8.

Στη συνέχεια δοκιμάστηκε να γίνει εκπαίδευση και έλεγχος πάνω στα πλήρη δεδομένα με χρήση του μέσου των confidences και βρέθηκε ποσοστό επιτυχίας 95.22%. Ποσοστό λίγο καλύτερο απο αυτό του βέλτιστου ταξινομητή. Πράγματι λοιπόν ο συνδιασμός των ταξινομητών δημιουργεί έναν καλύτερο συνολικό ταξινομητή. Η πρόταση που έχουμε να κάνουμε εδώ αφορά τα confidence scores. Απο τη φάση validation έχουμε κρατήσει τους πίνακες σύγχυσης για κάθε περίπτωση. Οι

στήλες του πίνακα σύγχυσης μας φανερώνουν τα ψηφία που κατηγοριοποιήθηκαν ώς την κατηγορία αυτής της στήλης. Οι κανονικοποιημένες στήλες μας δίνουν σε έναν βαθμό το confidence για κάθε ψηφίο καθώς αν στην στήλη 1 υπάρχουν στοιχεία στην 3η γραμμή σημαίνει οτι ταξινομήσαμε ως «μηδέν» κάποιο ψηφίο που ήταν «δύο». Έτσι μπορούμε να εξάγουμε συνολικά confidences για κάθε ψηφίο για κάθε ταξινομητή. Απο τη φάση validation έχουμε ήδη κρατήσει τους confidence matrices οπότε η διαδικασία είναι εύκολη και στη συνέχεια θα γίνει εκτίμηση πάνω στα πλήρη δεδομένα.

Τελικά τα αποτελέσματα δεν ήταν καλύτερα απο ότι είχαμε μέχρι στιγμής. Συγκεκριμένα πάνω στα πλήρη δεδομένα βρήκαμε βέλτιστη απόδοση με χρήση του μέσου ίση με 94.87%.

Στη συνέχεια με βάση το validation και train set, εξήγαμε τα confidence scores και τα συνδιάσαμε με τις κλάσεις των ψηφίων του validation set ωστε να χρησιμοποιήσουμε έναν γραμμικό ταξινομητή. Έπειτα για κάθε δείγμα του test set θα εξάγουμε τα confidence scores και θα προχωρήσουμε σε ταξινόμιση του νέου δείγματος αυτού. Για την ταξινόμιση αυτή χρησιμοποιήσαμε lda ανάλυση με χρήση της συνάρτησης classify. Το αποτέλεσμα βρέθηκε 95.02% το οποίο δεν είναι ικανοποιητικό καθώς είναι μικρότερο απο το ήδη υπάρχων ποσοστό. Ωστόσο είναι καλύτερο απο την απόδοση του lda ταξινομητή πάνω στα πλήρη δεδομένα. Στη συνέχεια όμως εκπαιδεύσαμε εκ νέου Support Vector Machines και το ποσοστό βρέθηκε 95.32%, η οποία είναι αρκετά καλή απόδοση.

Έπειτα θα χρησιμοποιήσουμε linear regression, λόγω της υπόδειξης, ωστε να προβούμε σε γραμμικό συνδιασμό των features και να βρούμε τα κατάλληλα βάρη-συντελεστές του υπερεπιπέδου σε αυτόν τον χώρο των 30 διαστάσεων, που περιγράφει τις σχέσεις μεταξύ features και κλάσεων. Έτσι αν έχουμε το training example $\mathbf{x_i}$ με κλάση $\mathbf{y_i}$ θα γράψουμε το πρόβλημα εύρεσης των συντελεστών σε matrix μορφή και θα επιλύσουμε το πρόβλημα με normal equations.

$$\begin{bmatrix} 1 & x_1^1 & x_1^2 & \cdots & x_1^{30} \\ 1 & x_2^1 & x_2^2 & & x_2^{30} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n^1 & x_n^2 & \cdots & x_n^{30} \end{bmatrix} = X \quad , \quad \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = b$$

Άρα ψάχνουμε τους κατάλληλους συντελεστές ωστε

$$Xw = b \Rightarrow w = (X^T X)^{-1} X^T b = X^+ b$$

Επιλύοντας την παραπάνω σχέση βρίσκουμε τα βάρη w και τελικά ένα καινούργιο δείγμα s θα πάρει την τιμη

$$\hat{y} = s^{\mathrm{T}} w$$

Επειδή αυτή η τιμή δεν θα είναι ακέραιος αριθμός θα την αντιστοιχήσουμε στον πιο κοντινό ακέραιο στο πεδίο [0,9] οπού πρέπει να προσέξουμε να προσθέσουμε μια μονάδα σαν αρχικό feature σε κάθε sample. Τα αποτελέσματα με αυτή τη μέθοδο είναι απογοητευτικά, καθώς αρκετές εκτιμήσεις παίρνουν τιμές κάτω του μηδενός.

Χρειάζεται λοιπόν περισσότερη μελέτη ωστε να εκπαιδεύσουμε σωστά αυτά τα νέα χαρακτηριστικά. Επίσης λόγω του οτι οι ταξινομητές μας δεν επέστρεφαν κάποιο confidence score θα χρειαστεί περισσότερη μελέτη ωστε να βρεθούν πιο σωστά confidences για κάθε ταξινομητή.