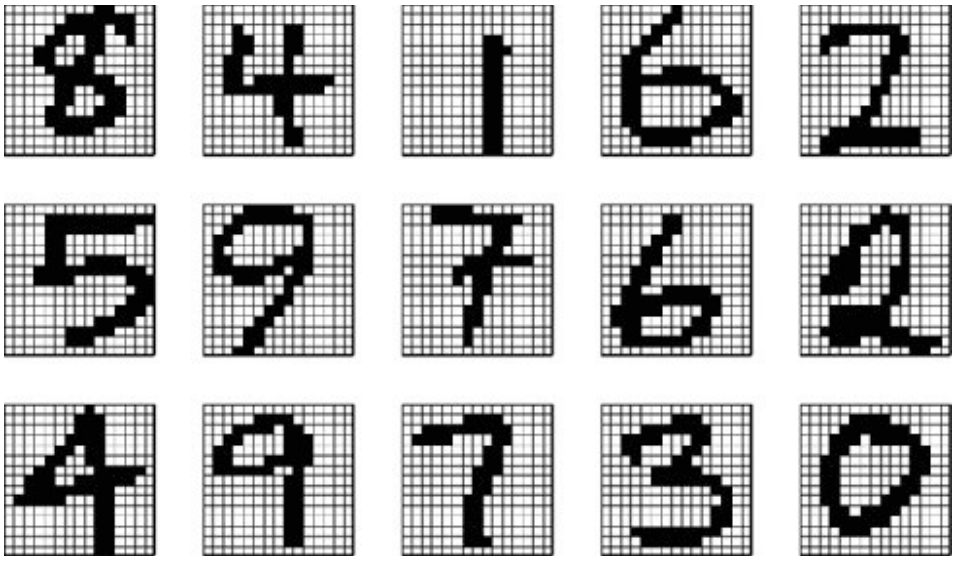
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

1η Εργαστηριακή Άσκηση

Οπτική Αναγνώριση Ψηφίων

Μάθημα : Αναγνώριση Προτύπων



Ροή Σ

Συνεργάτες :

- Βαβουλιώτης Γεώργιος (**Α.Μ. : 03112083**)
- Σταυρακάκης Δημήτριος (**Α.Μ. : 03112017**)

Σκοπός: Όπως αναφέραμε και στην προπαρασκευή της άσκησης αυτής, σκοπός μας είναι να υλοποιήσουμε ενα σύστημα οπτικής αναγνώρισης ψηφίων. Η συνέχεια λοιπόν της προπαρασκευής, περιλαμβάνει την υλοποίηση και σύγκριση αρκετών διαφορετικών ταξινομητών καθώς και τον σχολιασμό των αποτελεσμάτων του καθενός ξεχωριστά αλλά και τελικά συμπεράσματα ώστε να γίνει προφανές ποιός ταξινομητής είναι καλύτερος για το πρόβλημα της οπτικής αναγνώρισης ψηφίων.

Εκτέλεση Άσκησης

Βήμα 10 : Στο βήμα αυτό καλούμαστε να υπολογίσουμε τις a-priori πιθανότητες των ψηφίων. Είναι προφανές οτι για τον υπολογισμό των a-priori θα χρησιμοποιήσουμε τα train δεδομένα. Συγκεκριμένα η apriori πιθανότητα για κάθε ψηφίο προκύπτει απο το λόγο του πλήθους εμφάνισης κάθε ψηφίου προς το μέγεθος των train data, δηλαδή :

$$Pr[C_k] = \frac{\textit{πλήθος εμφανίσεων digit k}}{\textit{πλήθος train data}}$$

Προφανώς τις a-priori πιθανότητες τις κρατάμε σε ενα πίνακα αφού θα μας φανούν χρήσιμες στη συνέχεια.

Βήμα 11 : Στο βήμα αυτό καλούμαστε να υλοποιήσουμε ενα Bayesian ταξινομητή για να κάνουμε την ταξινόμηση των test δεδομένων σε μια απο τις 10 κατηγορίες. Στη προσπάθεια μας αυτή, θα χρησιμοποιήσουμε τις μέσες τιμές και τις διασπορές που υπολογίσαμε στο ερώτημα 7 της προπαρασκευής. Αρχικά θα πρέπει να πούμε οτι υλοποιήσαμε με τη βοήθεια του Matlab ενα Gaussian Naive Bayesian Classifier(υποθέτουμε δηλαδή οτι τα digits ακολουθούν κανονική κατανομή με μέση τιμή και διασπορά αυτές του βήματος 7), ο οποίος θα κατατάσσει κάθε ψηφίο στη παρακάτω κατηγορία, μεγιστοποιώντας στην ουσία την posterior πιθανότητα :

$$y = argmax_k Pr[C_k]\cdot Pr[x \mid C_k]$$

Επειδή έχουμε ανεξαρτησία χαρακτηριστικών μπορούμε να χρησιμοποιήσουμε τον λογάριθμο της έκφρασης μέσα στο argmax όπως είχαμε αναφέρει και στις διαλεξεις του μαθήματος και προκύπτει :

$$y = argmax_k log(Pr[C_k]\cdot Pr[x \mid C_k]) = argmax_k log\left(\prod_{i=1}^{N}\left(Pr[C_{k_i}]\cdot Pr[x \mid C_{ki}]\right)\right) = argmax_k\left(\sum_{i=1}^{N} log(Pr[x \mid C_{ki}]) + log\left(Pr[C_{k_i}]\right)\right)$$

όπου $Pr[x \mid C_{ki}] = \frac{1}{\sqrt{2\pi\sigma^2}}\cdot e^{-\frac{(x-\mu)2}{2\sigma 2}}$ και N =256.

Επομένως αυτή είναι η ιδέα με την οποία υλοποιήσαμε τον Bayesian ταξινομήτής μας. Το success ratio που μας έδωσε όπως αυτό φαίνεται στο Matlab είναι το εξής :

```
Naive Bayesian Success Ratio : 70.5531
```

Για πληρότητα σας παρουσιάζουμε και το success ratio για καθένα απο τα digits που κατάφερε ο ταξινομητής αυτός :

```
Naive Bayesian Success Ratio of 0
   80.7799

Naive Bayesian Success Ratio of 1
   96.5909

Naive Bayesian Success Ratio of 2
   61.1111

Naive Bayesian Success Ratio of 3
    50

Naive Bayesian Success Ratio of 4
   27.5000

Naive Bayesian Success Ratio of 5
    40

Naive Bayesian Success Ratio of 6
   91.7647

Naive Bayesian Success Ratio of 7
   87.0748

Naive Bayesian Success Ratio of 8
   66.2651

Naive Bayesian Success Ratio of 9
   87.0056
```

Βήμα 12 : Το βήμα αυτό είναι ίδιο με το βήμα 11 μόνο που πλέον καλούμαστε να υλοποιήσουμε ενα Bayesian ταξινομητή υποθέτοντας οτι η διασπορά πλέον είναι 1 για ολα τα χαρακτηριστικά. Η συλλογιστική πορεία που ακολουθήσαμε είναι ακριβώς η ίδια με το βήμα 11 και το αποτέλεσμα το οποίο πήραμε όσο αφορά το success ratio αυτού του Bayesian ταξινομητή φαίνεται παρακάτω :

```
Naive Bayesian (with Variance equal to 1) Success Ratio : 81.2656
```

Για πληρότητα σας παρουσιάζουμε και το success ratio για καθένα απο τα digits που κατάφερε ο ταξινομητής αυτός :

```
Naive Bayesian (with Variance equal to 1) Success Ratio of 0
   82.7298

Naive Bayesian (with Variance equal to 1) Success Ratio of 1
   98.1061

Naive Bayesian (with Variance equal to 1) Success Ratio of 2
   73.2323

Naive Bayesian (with Variance equal to 1) Success Ratio of 3
   78.9157

Naive Bayesian (with Variance equal to 1) Success Ratio of 4
    75

Naive Bayesian (with Variance equal to 1) Success Ratio of 5
   76.2500

Naive Bayesian (with Variance equal to 1) Success Ratio of 6
   84.1176

Naive Bayesian (with Variance equal to 1) Success Ratio of 7
   79.5918

Naive Bayesian (with Variance equal to 1) Success Ratio of 8
   79.0960

Naive Bayesian (with Variance equal to 1) Success Ratio of 9
   79.0960
```

Σχολιασμός αποτελεσμάτων Βημάτων 11 και 12 : Παρατηρούμε οτι ο δεύτερος Bayesian Classifier έχει μεγαλύτερο success ratio, με άλλα λόγια βλέπουμε οτι δίνει καλύτερα αποτελέσματα εκείνος ο ταξινομητής ο οποίος δεν λαμβάνει υπόψην του την πραγματική διασπορά αλλά εκείνος που την θεωρεί παντού ίση με 1. Αυτό γίνεται ο ταξινομητής του βήματος 12 έχει μεγάλη απόκλιση απο τις μέσες τιμές άρα κάνουμε καλή πρόβλεψη επειδή έχουμε κανονική κατανομή και όπως είναι γνωστό το 95% των δειγμάτων βρίσκονται στο διάστημα (m-v, m+v). Θα πρέπει να επισημάνουμε οτι στο Βήμα 11 στη διασπορά προσθέσαμε ενα μικρό παράγοντα ώστε να μην υπάρχουν διασπορές χαρακτηριστικών ίσες με το 0 για να πάρουμε ενα καλύτερο αποτέλεσμα. Εν κατακλείδι αυτό που καταλαβαίνουμε είναι οτι δεν είναι πάντα καλό να λαμβάνουμε υπόψην μας τις πραγματικές τιμές της διασποράς για να κάνουμε καλό classification.

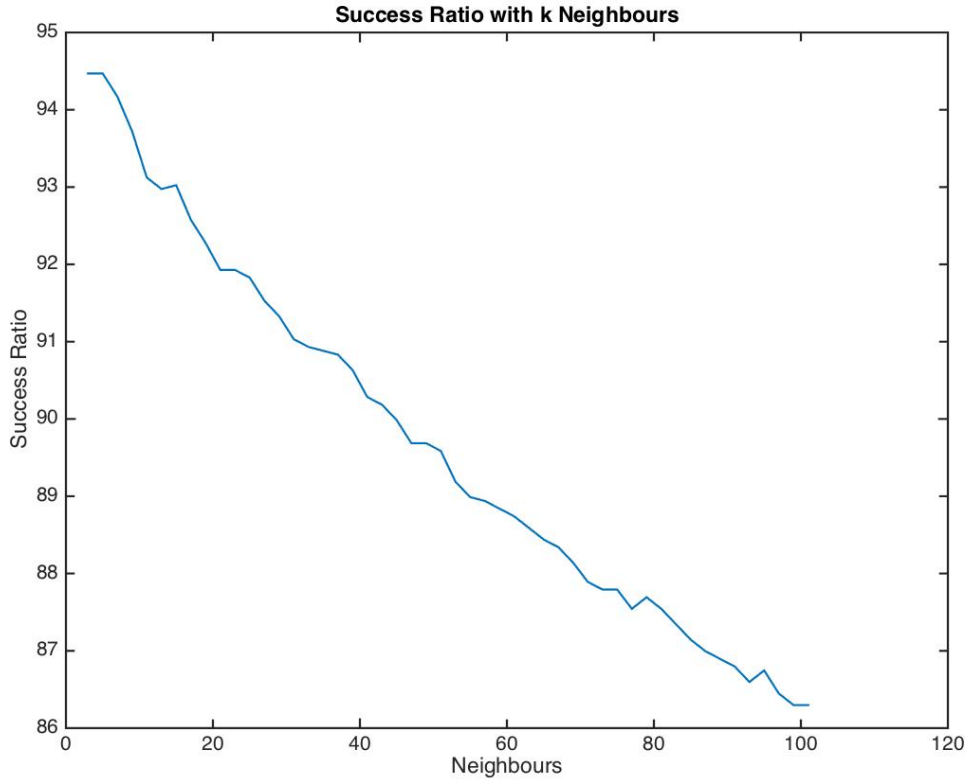
Βήμα 13 : Στο βήμα αυτό καλούμαστε να υλοποιήσουμε τον αλγόριθμο του κοντινότερου γείτονα-1(NNR-1), για να κάνουμε classification για τα 100 digits των test δεδομένων. Η ιδέα που υλοποιούμε στο Matlab είναι οτι για καθένα απο τα 100 ψηφία των test data υπολογίζουμε τις αποστάσεις απο τα train δεδομένα. Βρίσκοντας την ελάχιστη απόσταση απο αυτές για κάθε digit καταφέρνουμε να κατατάξουμε το κάθενα digit των test δεδομένων στην κατηγορία που ανήκει το ψηφίο των train data απο το απέχει λιγότερο.
Το success ratio που πήραμε με χρήση του NNR-1 χρησιμοποιώντας 100 test data και 1000 train είναι το παρακάτω :

```
NNR1 Ratio:
    90
```

Βήμα 14 : Στο Βήματα 14(α),14(β) αυτό που έχουμε να κάνουμε είναι να ταξινομήσουμε όλα τα test δεδομένα με χρήση όλων των train δεδομένων με την βοήθεια του αλγορίθμου NNR-1. Η συλλογιστική που ακολουθήσαμε είναι η ίδια, με την εξαίρεση οτι πλέον έχουμε να κάνουμε με περισσότερα δεδομένα test και train.
Το success ratio που πήραμε με χρήση του NNR-1 χρησιμοποιώντας όλα τα test και train data είναι το παρακάτω :

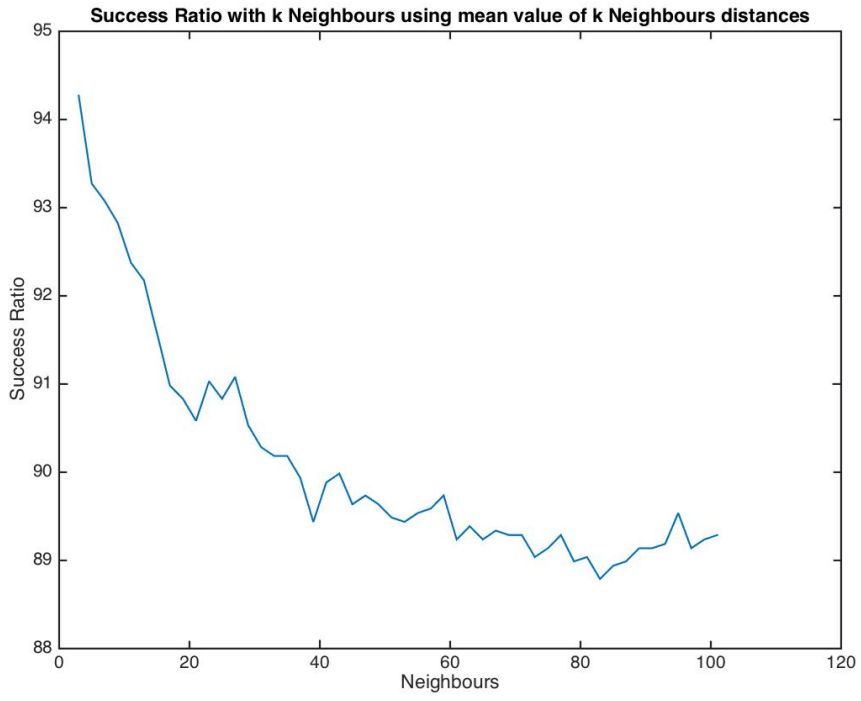
```
NNR1 Ratio of Total Classification:
   94.3697
```

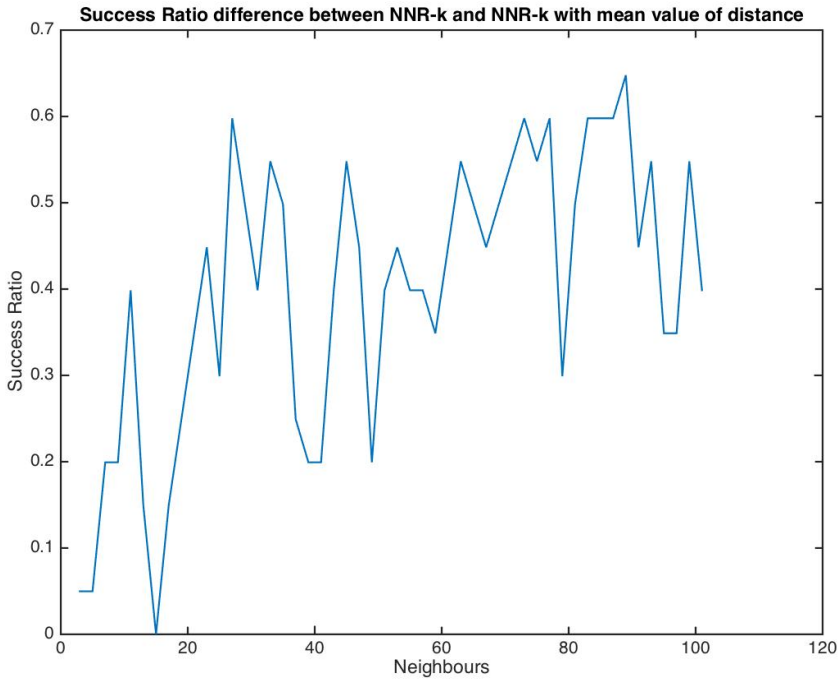
Στο βήμα14(γ) χρησιμοποιούμε τον αλγόριθμο NNR-k δηλαδή για κάθε test δεδομένο βρίσκουμε τους k κοντινότερους γείτονες του απο τα train δεδομένα και το κατατάσσουμε στην κατηγορία εκείνη στην οποία ανήκουν οι περισσότεροι απο τους k γείτονες του. Ενδεικτικά βάλαμε το k να πηγαίνει απο 3 με βήμα 2(αφού θέλουμε περιτές τιμές του k) μέχρι την τιμή 101 και δημιουργήσαμε το παρακάτω διάγραμμα το οποίο δείχνει το success ratio για τις διάφορες τιμές του k:



Σχολιασμός : Απο το παραπάνω διάγραμμα είναι προφανές οτι όσο αυξάνεται το k τόσο το success ratio μειώνεται.

Στη συνέχεια, στο Βήμα 14(δ) αυτό που καλούμαστε να κάνουμε είναι να βελτιώσουμε τον αλγόριθμο NNR-k. Η ιδέα που υλοποιήσαμε στο σημείο αυτό είναι η εξής: Αντί να επιλεχθεί αυθαίρετα η κλάση στην οποία ανήκουν οι περισσότεροι, επιλέγεται η κλάση στην οποία εμφανίζεται μικρότερη μέση τιμή απόστασης των k γειτόνων από το test δεδομένο. Στη συνέχεια παρατίθενται συγκριτικά διαγράμματα της βελτιωμένης έκδοσης του αλγορίθμου σε σχέση με τον κλασσικό NNR-k αλγόριθμο(το πρώτο δείχνει το success ratio της βελτιωμένης έκδοσης καθώς αυξάνεται το k και το δεύτερο είναι η σύγκριση αυτής με τον κλασσικό αλγόριθμο καθώς αυξάνεται το k) :





Σχολιασμός : Απο το πρώτο διάγραμμα βλέπουμε εκ νέου οτι όταν αυξάνεται το k μειώνεται το success ratio. Απο το δεύτερο διάγραμμα γίνεται αντιληπτό ότι η βελτίωση των μέσων τιμών είναι αισθητά καλύτερη από τον κλασσικό αλγόριθμο όταν ληφθούν υπόψην αρκετοί γείτονες.όπου φαίνεται και πάλι ότι αύξηση του k μειώνει το ποσοστό επιτυχίας.

Βήμα 15 : Στο βήμα αυτό καλούμαστε να υλοποιήσουμε 2 ακόμα ταξινομητές στους οποίους θα πρέπει να συμπεριλάβουμε τα SVM, τα οποία είναι είναι μοντέλα τα οποία εκτός των άλλων χρήσεων τους μπορούν να χρησιμοποιηθούν και ως πιθανοτικοι ταξινομητές. Η ιδέα πάνω στην οποία στηρίζονται τα SVMs είναι οτι τοποθετούν ενα δεδομένο σε μια απο τις δυο κατηγορίες που τους δίνονται μεγιστοποιώντας την απόσταση απο την επιφάνεια απόφασης. Αυτό που εμείς υλοποιήσαμε με τη βοήθεια του Matlab είναι 10 SVM ταξινομητές, ενας για κάθε digit, όπου ο καθένας εξετάζει αν ένω ψηφίο ανήκει στην κατηγορία για την οποία εκείνος είναι υπεύθυνος. Όπως αναφέρεται και στην εκφώνηση της άσκησης χρησιμοποιήσαμε δυο είδη πυρήνων, ενας γραμμικός και ενας πολυωνιμικός άρα συνολικά 20 SVM ταξινομητές.

Όπως είναι όμως προφανές υπάρχει και η περίπτωση κάποιο ψηφίο να κατατάσσεται απο 2 ή περισσότερους ταξινομητές στην κλάση τους. Στο σημείο αυτό εμείς θα πρέπει να είμαστε σε θέση να αποφασίσουμε σε ποιά κλάση τελικά θα ταξινομηθεί ο ψηφίο αυτό. Ο διαχωρισμός έγινε με χρήση ενος score για κάθε κλάση και τελικά επιλογή της κλάσης με το μικρότερο score. Η εκπαίδευση των ταξινομητών έγινε με χρήση της συνάρτησης svmtrain() του matlab, στην οποία δώσαμε σαν ορίσματα τα train data, πίνακα για την κατάταξη των κλάσεων, το είδος του πυρήνα και για options = max_iter όπως θεωρήσαμε κατάλληλο βάση των doc του mathworks. Μετά την εκπαίδευση, χρησιμοποιούμε την συνάρτηση mysvmclassify(τροποποιημένη εκδοχή της Built-in svmclassify του matlab) ώστε αυτή να μας επιστρέφει και το score που χρειαζόμαστε για την κατηγοριοποίηση των ψηφίων. Η πηγή από την οποία βοηθηθήκαμε υπάρχει στη function mysvmclassify. Αφού λοιπόν τα κάναμε όλα αυτά υπολογίσαμε τα κατάλληλα ratios που μας δίνουν αυτοι οι ταξινομήτες.

Το success ratio για τον γραμμικό SVM ταξινομητή είναι το εξής :

```
Svm with Linear Kernel Success Ratio : 89.9352
```

Το success ratio για τον πολυωνιμικό SVM ταξινομητή είναι το εξής :

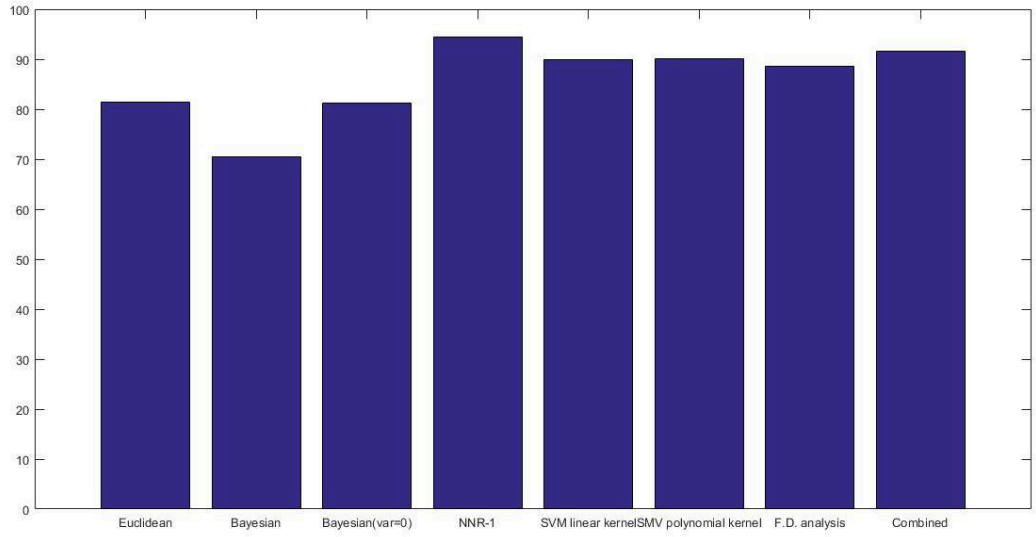
```
Svm with Polynomial Kernel Success Ratio : 90.0847
```

Ο έταιρος ταξινομητής που χρησιμοποιήσαμε είναι ο Fit discriminant analysis classifier. Αυτό επιτεύχθηκε με τη βοήθεια της fitcdiscr() του Matlab σε συνδυασμο με την predict() (η αντίστοιχη αναφορά στο mathworks υπάρχει επίσης στο script). Υπολογίσαμε και τα success ratio αυτου του ταξινομητή. Το success ratio για τον Fit discriminant analysis classifie είναι το εξής :

```
Fit discriminant analysis classifier Success Ratio : 88.5401
```

Τα success ratio per digit για καθένα απο τους παραπάνω ταξινομητές(svm liner, svm polynimial, fit discriminant) υπάρχουν στο zip αρχείο που σας παραδίδω σε mat files με όνοματα αντίστοιχα των ταξινομητών παραπάνω, για να μην τα βάλω και αυτά με screenshot και γίνει πολύ μεγάλη η αναφορά.

Βήμα 16 : Στο πρώτο ερώτημα του βήματος αυτού καλούμαστε να συνδυάσουμε διάφορους απο τους ταξινομητές τους οποίους έχουμε υλοποιήσει ώστε να καταφέρουμε να πετύχουμε καλύτερο αποτέλεσμα. Στη συνέχεια σας παρουσιάζω ενα διάγραμμα το οποίο δείχνει το total success ratio κάθενα απο τους ταξινομητές που θα χρησιμοποιήσουμε για να φτιάξουμε ενα συνδυασμό απο ταξινομητές :



Η λογική με την οποία θα συνδυάσουμε τους ταξινομητές είναι οτι εμπιστευόμαστε κάθε φορά την πλειοψηφία των κατηγοριοποιήσεων για να καταφέρουμε να πάρουμε καλύτερο αποτέλεσμα απο τα επιμέρους. Θα πρέπει να τονιστεί οτι σε περίπτωση απουσίας πλειοψηφίας εμπιστευόμαστε τα αποτελέσματα εκείνων των ταξινομητών με το μεγαλύτερο success ratio(αυτό το κάναμε για να είμαστε πλήρως σωστοί, ωστόσο βρήκαμε οτι οι ισοπαλίες είναι 10 το πλήθος κάτι το οποίο σημαίνει οτι ο μη χειρισμός τους δεν θα χαλούσε κατά πολύ το αποτέλεσμα). Τέλος ο combo classifier που φτιάξαμε πετυχαίνει success ratio ίσο με :

```
Combining classifiers Success Ratio : 91.6293
```

Παρατηρουμε ότι δεν είναι και το καλύτερο δυνατό αποτέλεσμα αφού με μια γρήγορη ματιά βλέπουμε οτι ο NNR-1 δίνει οριακά καλύτερα αποτελέσματα απο μόνος του. Το αποτέλεσμα αυτό είναι λογικό αν σκεφτεί κανείς οτι κάναμε συνδυασμό ταξινομητών που ενδεχομένως έχουν κοινά λάθη. Το success ratio per digit για τον combo classifier υπάρχει στο zip αρχείο που σας παραδίδω σε mat file με όνομα αντίστοιχ του ταξινομητή παραπάνω, για να μην τα βάλω και αυτά με screenshot και γίνει πολύ μεγάλη η αναφορά.
Για να καταφέρουμε καλύτερο αποτέλεσμα του combo classifier μπορούμε πλέον να χρησιμοποιήσουμε κάποιους απο τους NNR-k ταξινομητές, Συγκεκριμένα αν επιλέξουμε εκείνους με k = 3 και k = 9(ώστε και πάλι να έχω περιτό αριθμό ταξινομητών), το αποτέλεσμα που πήραμε όσο αφορά το success ratio για τον νέο combo classifier είναι 94,38% το οποίο προφανώς καλύτερο απο το προηγούμενο. Γενικά μπορούμε να πετύχουμε και πολύ καλύτερα αποτελέσματα αν κάνουμε καλύτερο συνδυασμό.