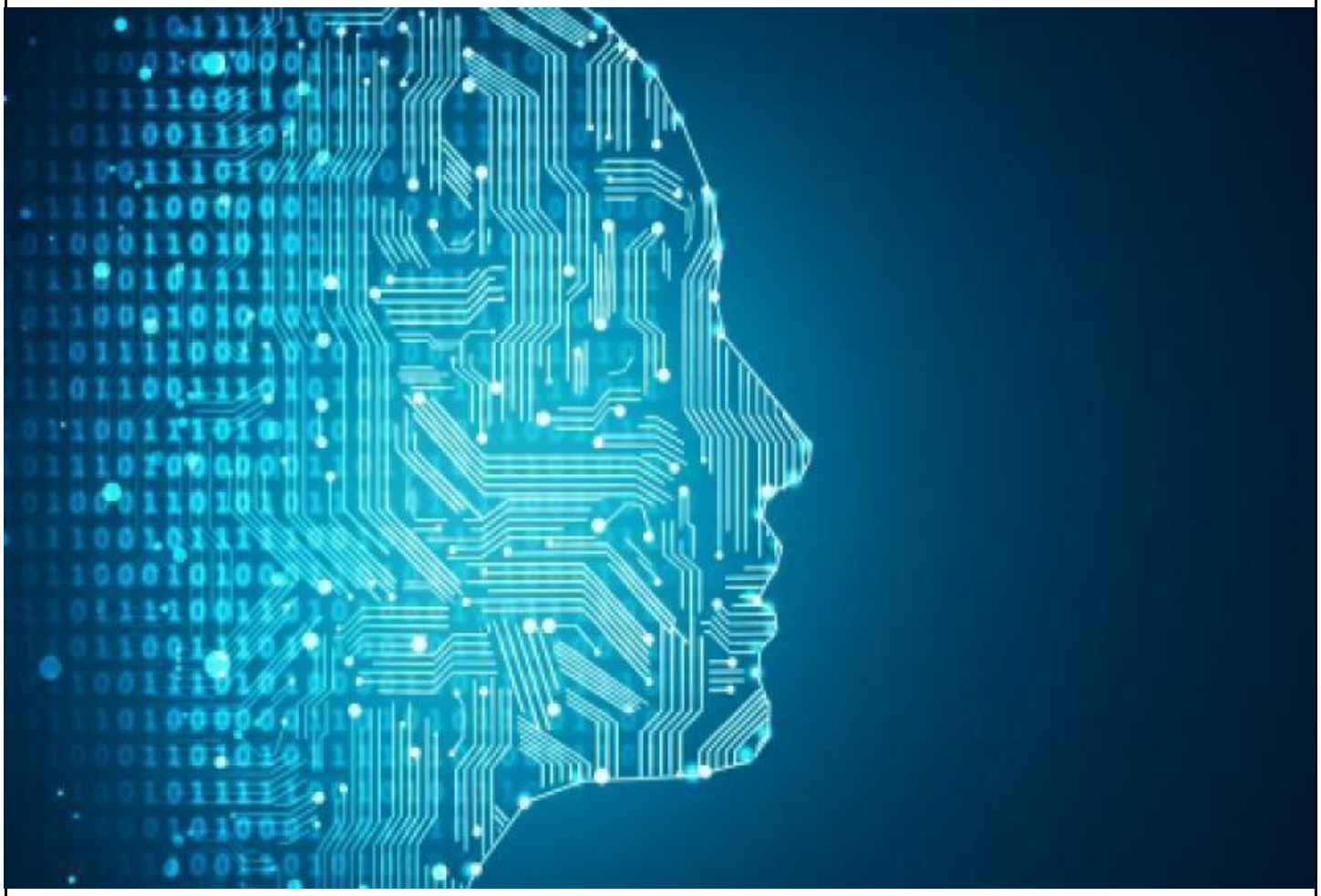


Αναγνώριση Προτύπων

Εργασία 4

Αλβανάκη Παρασκευή

ΑΜ 57286

ΕΡΩΤΗΜΑ 4.1

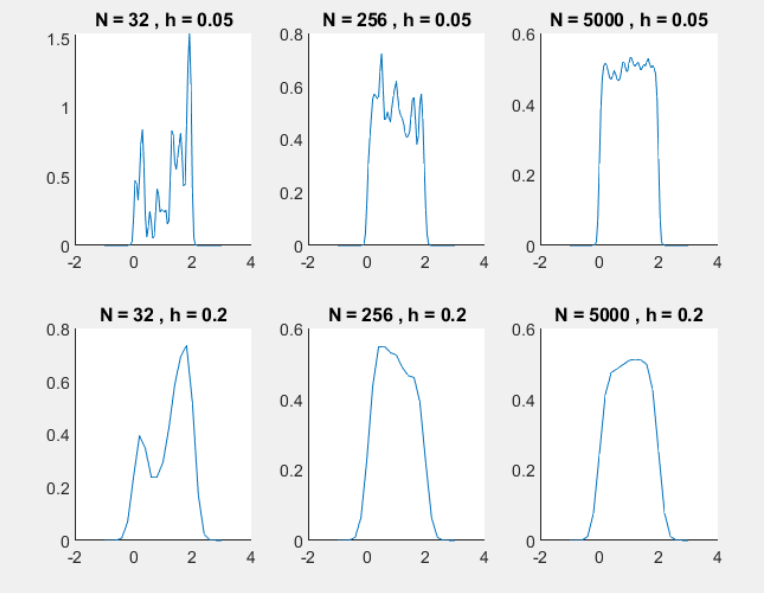
Α)

Αρχικά δημιουργούμε τυχαίες τιμές από 0 έως 2 για 32, 256 και 5000 σημεία αντίστοιχα. Στη συνέχεια προσπαθούμε να προσεγγίσουμε την pdf της κατανομής με την μέθοδο παραθύρων και την γκαουσιανή ως πυρήνα. Καλούμε τη συνάρτηση Parzen_gauss_kernel η οποία δέχεται ως παράμετρο το πλάτος h ,τα δείγματα και την ελάχιστη και μέγιστη τιμή για την οποία θέλουμε να υπολογίσουμε την pdf.Στη συνέχεια «κεντράρει» σε κάθε σημείο χ μια γκαουσιανή κατανομή ,τις αθροίζει(λαμβάνοντας υπ' όψην όσες βρίσκονται σε απόσταση μικρότερη ή ίση του h από το σημείο) και διαιρεί με τον αριθμό των δειγμάτων και τον όγκο(που στην περίπτωση μας επειδή είναι 1d κατανομή είναι h). Ουσιαστικά το γκαουσιανο kernel γίνεται shift κατά x_i (μ=xi) και scaled κατά h(σ=h).Σε κάθε επανάληψη μετακινείται το αριστερό άκρο που ξεκινάμε να μετράμε το σ προκειμένου σε κάθε σημείο να λαμβάνουμε υπ' όψην τα χ τα οποία είναι μέσα στα όρια του h.Η διαίρεση με το $\frac{1}{hN}$ γίνεται έτσι ώστε το συνολικό p(x) να ισούται με 1.

Ο κώδικας με τον οποίο υλοποιήθηκε το παραπάνω είναι ο εξής:

```
m = 0;
S = 1;
P = 1/2;
N = [32 256 5000];
for i=1:3
    X{i}=2*rand(1,N(i));
end
h = [0.05 0.2];
figure('name','1.a)');

for i = 1:3
    subplot(2,3,i);
    hold on;
    pdfx_approx = Parzen_gauss_kernel(X{i},h(1),-1,3);
    plot(-1:h(1):3,pdfx_approx);
    title(['N = ',num2str(N(i)),' , h = ',num2str(h(1))]);
    hold off;

    subplot(2,3,i+3);
    hold on;
    pdfx_approx = Parzen_gauss_kernel(X{i},h(2),-1,3);
    plot(-1:h(2):3,pdfx_approx);
    title(['N = ',num2str(N(i)),' , h = ',num2str(h(2))]);
    hold off;

end
```

Τα plot που λαμβάνουμε είναι τα εξής:



Παρατηρούμε πως για μικρότερο h λαμβάνουμε pdf με μεγαλύτερο θόρυβο το οποίο είναι λογικό καθώς λαμβάνονται υπ' όψην λιγότερα σημεία σε κάθε παράθυρο. Επιπλέον σε μεγάλα h ο υπολογισμός της pdf γίνεται σε λιγότερα σημεία με αποτέλεσμα μια πιο smooth καμπύλη. Όσον αφορά το πλήθος των σημείων που έχουμε στη διάθεση μας παρατηρούμε πως όσο αυξάνεται ο αριθμός των σημείων τόσο προσεγγίζεται καλύτερα η p(x) το οποίο είναι και το αναμενόμενο καθώς περισσότερα σημεία δίνουν καλύτερη προσέγγιση.

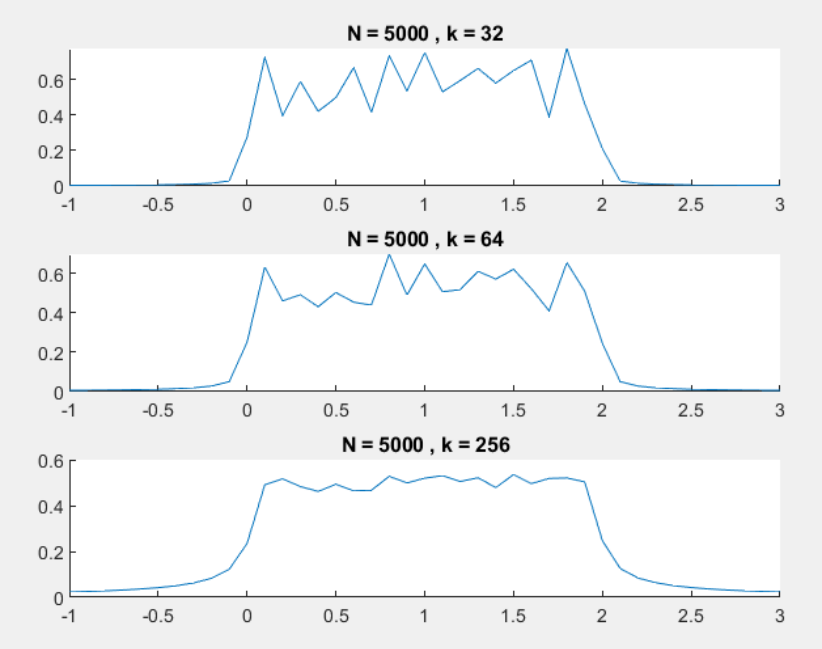
B)

Στον knn ταξινομητή ουσιαστικά για κάθε σημείο λαμβάνονται υπ' όψην οι κ γείτονες με την μικρότερη απόσταση. Σε αντίθεση με τον Parzen που είχε σταθερό παράθυρο, στον knn είναι σταθερός ο αριθμός των γειτόνων που λαμβάνονται υπ' όψην.Στον knn ταξινομητή πάλι κινούμαστε με ένα step για να κάνουμε loop through όλες τις τιμές ωστόσο σε κάθε σημείο x_i υπολογίζουμε την απόσταση του από όλα τα σημεία και επιλέγουμε ένα λάβουμε υπ' όψην τα k κοντινότερα. Ο τύπος για τον υπολογισμό της πιθανότητας είναι $\rho_n(x, w_i) = \frac{k_i/n}{V}$ με το k_i να είναι ο αριθμός προτύπων της κλάσης i στην περιοχή γύρω από το x, n ο συνολικός αριθμός προτύπων και v το πλάτος του παραθύρου το οποίο είναι μεταβλητό και εξαρτάται από τις αποστάσεις των κ γειτόνων από τα πρότυπα. Τέλος παρατηρούμε από το διάγραμμα παρακάτω ότι όσο μεγαλύτερος είναι ο αριθμός των γειτόνων τόσο καλύτερη είναι η προσέγγιση
Ο κώδικας με τον οποίο υλοποιήθηκε το παραπάνω είναι ο ακόλουθος.

```
%1.b
k = [32 64 256];

figure('name','1.b)');
for i = 1:3
    subplot(3,1,i);
    hold on;
    pdfx_approx_b = knn_density_estimate(X{3},k(i),-1,3,0.1);
    plot(-1:0.1:3,pdfx_approx_b);
    title(['N = ',num2str(N(3)),' , k = ',num2str(k(i))]);
    hold off;
```

```
end
```

Τα plot που δημιουργήθηκαν είναι τα παρακάτω:



ΕΡΩΤΗΜΑ 4.2

Α)

Για τη δημιουργία τυχαίων δειγμάτων χρησιμοποιήθηκε η συνάρτηση generate_gauss_classes η οποία παίρνει ως ορίσματα τα mean sigma και την πιθανότητα κάθε κατανομής και μέσω της mvnrnd δημιουργεί τα επιθυμητά χ και τα αντίστοιχα labels τους επιστρέφοντας 2 vectors 1 για τις τιμές και 1 για τα labels των τιμών αυτών. Έτσι δημιουργούμε 2 δείγματα 100 και 1000 προτύπων τα οποία αποτελούν το train και test σετ μας αντίστοιχα.

Ο κώδικας με τον οποίο υλοποιήθηκε το παραπάνω είναι ο εξής:

```
P = [0.5 0.3 0.2]';
m = [2 1 3];
Sigma(:,:,1) = 0.5;
Sigma(:,:,2) = 1;
Sigma(:,:,3) = 1.2;
randn('seed',0);
[X_train,train_label] = generate_gauss_classes(m,Sigma,P,100);
randn('seed',2);
[X_test,test_label] = generate_gauss_classes(m,Sigma,P,1000);
```

B)

Για k=1 k=2 και k=3 λαμβάνω τις ακόλουθες πιθανότητες λάθους.

The error is 0.482 for k = 1
The error is 0.408 for k = 2
The error is 0.392 for k = 3

Όπως είναι αναμενόμενο το μεγαλύτερο σφάλμα το έχω για k=1.Όταν έχουμε k=1 ουσιαστικά το χ λαμβάνει την τιμή του κοντινότερου γείτονα. Αυτό προφανώς δεν κανει generalize σχεδόν καθόλου με αποτέλεσμα το δείγμα του test set το οποίο περιέχει άγνωστα για τον knn πρότυπα να έχει μεγάλο σφάλμα. Αντίθετα όσο αυξάνουμε τον αριθμό k των γειτόνων που λαμβάνονται υπ' όψην έχουμε σαφώς καλύτερα αποτελέσματα με μικρότερη πιθανότητα σφάλματος.

Όσον αφορά τον Bayesian ταξινομητή γνωρίζουμε πως είναι ένα probabilistic μοντέλο και συνεπώς για κάθε άγνωστο δείγμα υπολογίζει το likelihood (για δεδομένα που ακολουθούν την gaussian κατανομή) και με βάση το likelihood της κάθε κλάσης το οποίο πολλαπλασιάζει με την prior πιθανότητα υπολογίζει μια τιμή ανάλογη του posterior του κάθε δείγματος.
The error with the Bayesian classifier is = 0.323

Στο συγκεκριμένο dataset παρατηρούμε πως ο Bayesian δίνει μικρότερο error από τον knn .Ωστόσο αυτό δεν ισχύει πάντα καθώς κάθε ταξινομητής έχει τα θετικά του και τα αρνητικά του και εξαρτάται από τα δεδομένα και τον τύπο του προβλήματος που προσπαθούμε να λύσουμε το αν ο Bayesian θα μας δώσει καλύτερο αποτέλεσμα.

Ο κώδικας με τον οποίο υλοποιήθηκε το παραπάνω είναι ο ακόλουθος.

```
% 2.b
for k=1:15
    X_knn{k} = k_nn_classifier(X_train,train_label,k,X_test);
    error_knn(k) = sum(X_knn{k}~=test_label)/length(test_label);
    disp([' The error is ',num2str(error_knn(k)),' for k =
',num2str(k)]);
end
X_knn_bayesian = bayes_classifier(m,Sigma,P,X_test);
error_bayesian =
sum(X_knn_bayesian~=test_label)/length(test_label);
disp(['The error with the Bayesian classifier is =
',num2str(error_bayesian)]);
```

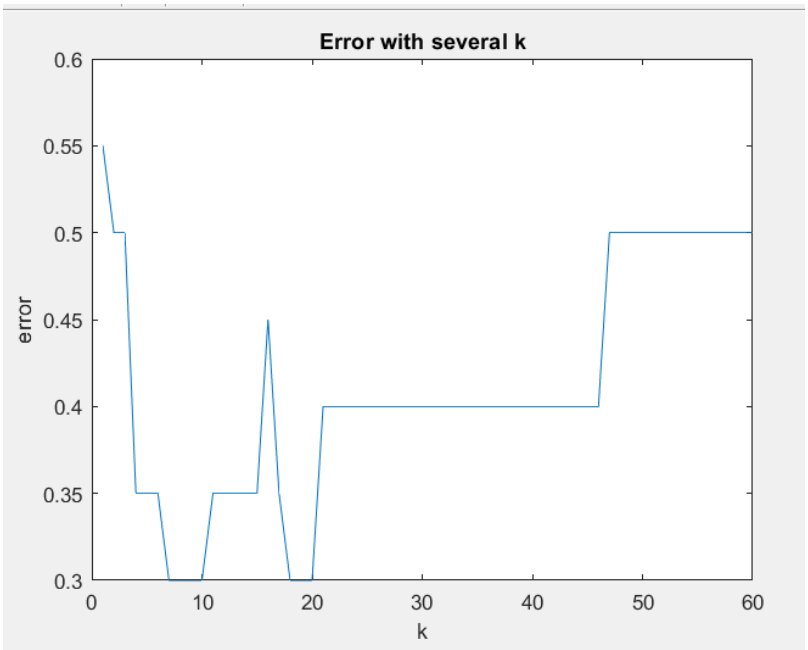
- Προαιρετικό β
  Για να δω ποιο είναι το βέλτιστο x θα πρέπει να δοκιμάσω διάφορες τιμές κ σε δεδομένα που δεν είναι στο training σετ ή στο test set καθώς αν ανήκουν στο training set είναι biased το αποτέλεσμα και το να ανήκουν στο τεστ σετ δεν είναι επιθυμητό καθώς στην πραγματικότητα test σετ είναι δεδομένα τα οποία θα πρέπει να χρησιμοποιούνται για την αξιολόγηση και όχι για βελτιστοποίηση. Επιλέγω λοιπόν το 20% των δειγμάτων του training να τα κάνω validation set και συνεπώς τα νέα training data να είναι το υπόλοιπο 80%.
  Δοκιμάζω διάφορες τιμές του κ και βρίσκω αυτήν με το μικρότερο validation error με τον παρακάτω κώδικα:

```
%2.b optional
x_train_n=[X_train(1:40) X_train(51:74) X_train(81:96)];
train_label_n=[train_label(1:40) train_label(51:74)
train_label(81:96)];

x_val_n=[X_train(41:50) X_train(75:80) X_train(97:100)];
val_label=[train_label(41:50) train_label(75:80)
train_label(97:100)];

for k=1:60
    label_knn{k} =
k_nn_classifier(x_train_n,train_label_n,k,x_val_n);
    pr_err_knn(k) =
sum(label_knn{k}~=val_label)/length(val_label);
end
[min_error, min_error_k] = min(pr_err_knn);
figure('name','4.2 B)');
plot(pr_err_knn);
title('Error with several k');
xlabel('k');
ylabel('error');
disp(['minimum error of the validation set is:
',num2str(min_error),' with k = ',num2str(min_error_k)]);
```

Έτσι παίρνω το αποτέλεσμα:

minimum error of the validation set is: 0.3 with k = 7

Το οποίο μπορώ να το δω και γραφικά:



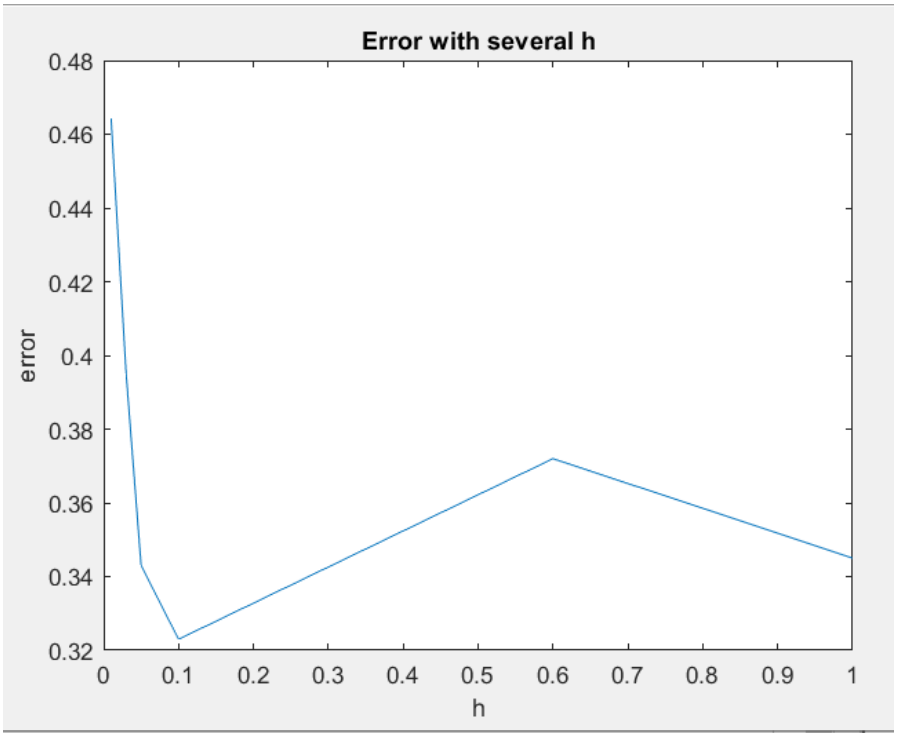
Αν τρέξω τον κώδικα για knn classification με κ=7 παίρνω test error :

The error is 0.331 for k = 7

Γ)

Στο ερώτημα αυτό παρατηρώ πως το μικρότερο error προκύπτει όταν h=0.1.Το h=0.1 είναι μια λογική τιμή για parzen window καθώς δεν είναι ούτε αρκετά μικρή για να λαμβάνει υπ' όψην ελάχιστα σημεία σε κάθε παράθυρο αλλά ούτε και αρκετά μεγάλη για να μην λαμβάνεται ικανοποιητικός αριθμός σημείων στην pdf.

Γραφική απεικόνιση δίνεται παρακάτω:

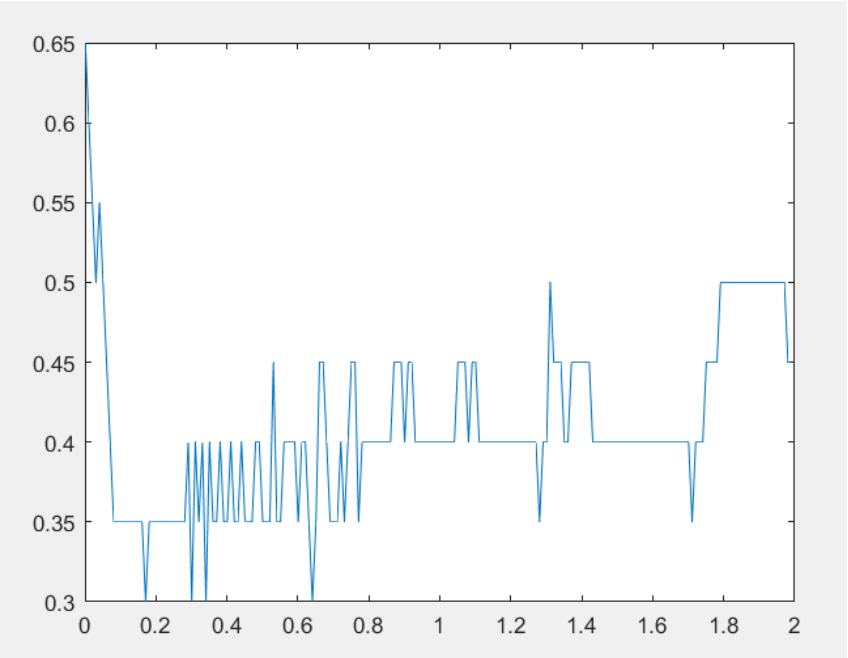


Ο κώδικας με τον οποίο υλοποιήθηκε το παραπάνω είναι ο εξής:

```
%2.c
h = [0.01 0.03 0.05 0.1 0.6 1 ];
for j=1:length(h)
    px1=0.5*Parzen_gauss_kernel(X_train(1:50),h(j),-4,10);
    px2=0.3*Parzen_gauss_kernel(X_train(51:80),h(j),-4,10);
    px3=0.2*Parzen_gauss_kernel(X_train(81:100),h(j),-4,10);
    [num,z]=max([ px1; px2; px3]);
    psi=ceil((X_test+4)/h(j));
    answer=z(psi);

    err_windows(j)=sum(answer~=test_label)/length(test_label);

    disp(['The error with Parzen Windows is :
',num2str(err_windows),' with a value of h = ',num2str(h(j))]);
end
```

```
figure('name','4.2 c)');
plot(h,err_windows);
title('Error with several h');
xlabel('h');
ylabel('error');
```

- Προαιρετικό γ
  Η διαδικασία είναι η ίδια με αυτή που ακολουθήθηκε στο προηγούμενο ερώτημα δηλαδή «σπάσιμο» του train set σε train και validation .Δοκιμάζοντας διάφορες τιμές του h και ελέγχοντας το error τους προκύπτει ότι το h με το μικρότερο validation error είναι το h=0.171 με validation error=0.3(υπάρχουν και αλλά 2 h τα οποία μας δίνουν το ίδιο validation error το 0.301 και το 0.641).Παρακάτω μπορούμε το δούμε και γραφικά:



Δ)

Για τις διάφορες τιμές του PNN προκύπτουν τα παρακάτω αποτελέσματα:

The error for PNN classifier with Parzen Windows is : 0.485 with a value of h = 0.01
The error for PNN classifier with Parzen Windows is : 0.352 with a value of h = 0.03
The error for PNN classifier with Parzen Windows is : 0.584 with a value of h = 0.05
The error for PNN classifier with Parzen Windows is : 0.571 with a value of h = 0.1
The error for PNN classifier with Parzen Windows is : 0.435 with a value of h = 0.6
The error for PNN classifier with Parzen Windows is : 0.5 with a value of h = 1

Παρατηρούμε πως οι τιμές του error είναι αρκετά υψηλές ανεξάρτητα από το αν το γκαουσιανο παράθυρο είναι μεγάλο ή μικρό .Συγκρίνοντας τις τιμές με αυτές του knn

ταξινομητή παρατηρούμε πως για μικρές τιμές του k οι οποίες δεν είναι οι βέλτιστες το error του knn είναι μικρότερο από αυτό του PNN πάλι για μη βέλτιστες τιμές.
Ο κώδικας με τον οποίο υλοποιήθηκε το παραπάνω είναι ο ακόλουθος.

```
%2.d
disp("2d");
net=parzenPNNlearn(X_train,train_label);
for j=1:length(h)
    [class,score,scores]=parzenPNNclassify(net,X_test,h(j));
    err_ParzenPNN=sum(class~=test_label)/length(test_label);

    disp(['The error for PNN classifier with Parzen Windows is :
',num2str(err_ParzenPNN),' with a value of h = ',num2str(h(j))]);
end
```