Μηχανική Μάθηση

3ο Σετ Ασκήσεων

ΟΝΟΜΑ: Βασίλειος
ΕΠΩΝΥΜΟ: Καλαϊτζόπουλος
ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ: 1066670
ΕΤΟΣ ΦΟΙΤΗΣΗΣ: 4ο
ΤΜΗΜΑ: Ηλεκτρολόγων Μηχανικών & Τεχνολογίας Υπολογιστών

Πρόβλημα 3.1

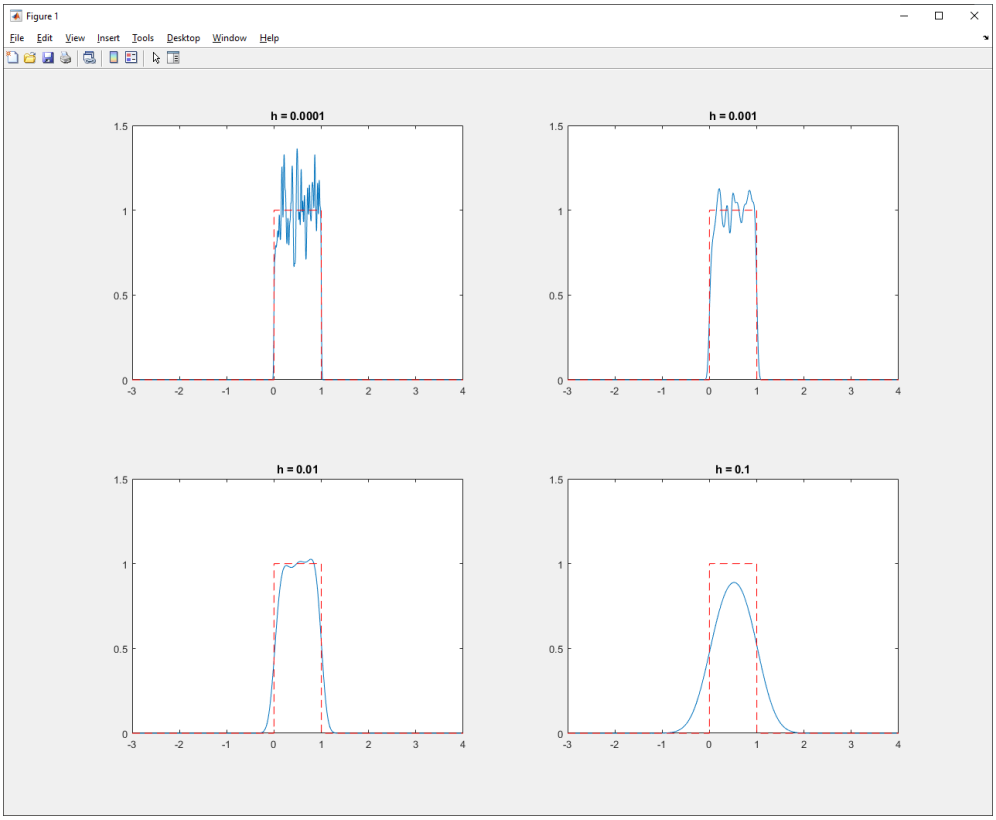
Στο συγκεκριμένο πρόβλημα θέλουμε να αξιολογήσουμε, πώς με την μέθοδο Kernel μπορούμε να προσεγγίζουμε πυκνότητες πιθανότητας. Έτσι θα δημιουργήσουμε 1000 υλοποιήσεις μιας τυχαίας ομοιόμορφα κατανεμημένης μεταβλητής στο διάστημα [0, 1] και θα προσεγγίσουμε την πυκνότητα πιθανότητας της χρησιμοποιώντας το Gaussian kernel που φαίνεται παρακάτω:

$$K(x,h) = \frac{1}{\sqrt{2\pi h}} e^{-\frac{1}{2h}x^2}$$

Έχοντας τις χίλιες υλοποιήσεις (N = 1000) τις τυχαίας μεταβλητής με την χρήση του παρακάτω τύπου προσεγγίζουμε την πυκνότητα πιθανότητας:

$$\hat{f}(x) = \frac{K(x - x_1, h) + \cdots + K(x - x_N, h)}{N}$$

όπου x_n είναι οι χίλιες υλοποιήσεις που δημιουργήσει. Επιπλέον δημιουργούμε και ένα διάνυσμα x το όπως φαίνεται και ενότητα ΚΩΔΙΚΕΣ (x_axis) το οποίο είναι η τιμές για τις οποίες υπολογίζουμε την τυχαία μεταβλητή. Έγιναν δοκιμές για προσέγγιση της πυκνότητας πιθανότητας για h = [0.0001, 0.001, 0.01, 0.1] τα αποτελέσματα των οποίων φαίνονται παρακάτω:



Από τα αποτελέσματα φαίνεται ότι όσο μικρότερο είναι το h έχουμε μια καλύτερη προσέγγιση στην βάση της f(x) ωστόσο στην κορυφή φαίνεται ότι δεν προσεγγίζουμε καλά τις τιμές της καθώς έχουμε πολλές κορυφές, πράγμα που συμβαίνει στην περίπτωση όπου h = 0.001. Επιπλέον παρατηρούμε ότι καθώς αυξάνεται το h συμβαίνει το αντίθετο, δηλαδή έχουμε καλύτερη προσέγγιση των τιμών της f(x) αλλά όχι τόσο του πεδίου ορισμού. Από τα παραπάνω παραδείγματα η καλύτερη προσέγγιση φαίνεται να είναι αυτή για h = 0.01.

Πρόβλημα 3.2

Στο συγκεκριμένο πρόβλημα θέλουμε να δημιουργήσουμε ένα classifier ο οποίος θα μπορεί να διακρίνει μεταξύ δύο συνόλων από δισδιάστατα διανύσματα τα οποία αντιστοιχούν σε ένα σημείο στο δισδιάστατο επίπεδο και έχουν label είτε star είτε circle. Θέλουμε να χρησιμοποιήσουμε την μέθοδο με τα kernel για να βρούμε το διαχωριστικό σύνορο γι' αυτό και αντιστοιχίζουμε την το αριθμητικό label ''1'' στο σύνολο stars και το ''-1'' σύνολο circles.

Θέλουμε να βρούμε έναν μετασχηματισμό Φ(X) όπου $X = [x_1, x_2]^T$ τον οποίο όταν τον εφαρμόζουμε στα σημεία θα το αντιστοιχεί στην σωστή ''ετικέτα''. Για να βρούμε τον μετασχηματισμό Φ(X) λύνουμε το παρακάτω πρόβλημα βελτιστοποίησης:

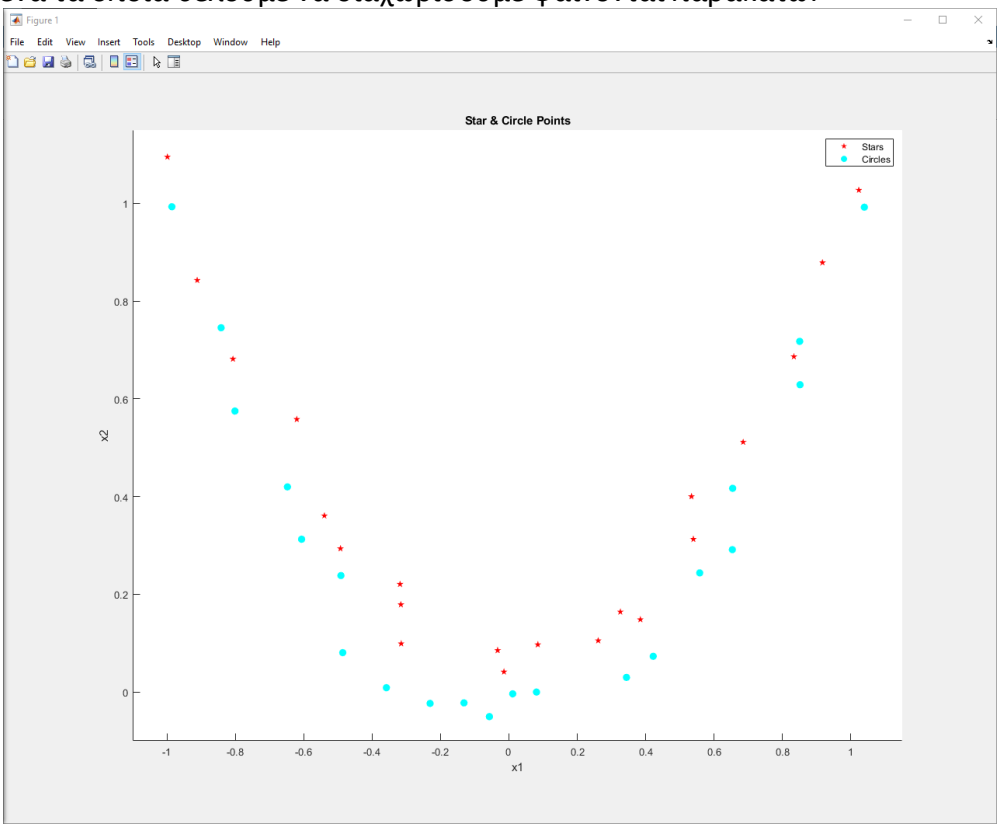
$$\min_{\Phi \in V} \left\{ \sum_{X_i \in stars} \left(1 - \Phi(X_i)\right)^2 + \sum_{X_j \in circles} \left(1 - \Phi(X_j)\right)^2 + \lambda \|\Phi(X)\|^2 \right\} =$$

$$= \min_{\Phi \in V} \left\{ \sum_{i=1}^{N} \left(Y_i - \Phi(X_i)\right)^2 + \lambda \|\Phi(X)\|^2 \right\}$$

όπου N είναι το πλήθος των διανυσμάτων των δύο συνόλων, N = N_stars + N_circles και V είναι ο διανυσματικός χώρος των συναρτήσεων που ορίζονται με την βοήθεια του παρακάτω Gaussian kernel:

$$K(X,Y) = e^{-\frac{1}{h}\|X-Y\|^2} = e^{-\frac{1}{h}\{(x_1-y_1)^2+(x_2-y_2)^2\}}$$

Τα δεδομένα τα οποία θέλουμε να διαχωρίσουμε φαίνονται παρακάτω:



Ερώτημα α)

Θέλουμε να αποδείξουμε μέσω του Representer Theorem ότι μπορούμε στα πρώτα δύο αθροίσματα να αντικαταστήσουμε το $\Phi(X)$ με την ορθογώνια προβολή του $\widehat{\Phi}(X)$ πάνω στον γραμμικό υποχώρο που δημιουργούν οι συναρτήσεις $K(X, X_i)$ όπου $X_i \in stars$ και η $K(X, X_j)$ όπου $X_j \in circles$ δηλαδή:

$$\widehat{\Phi}(X) = \sum_{X_i \in stars} \alpha_i K(X, X_i) + \sum_{X_j \in stars} \beta_j K(X, X_j)$$

Αρχικά αντιστοιχίζουμε τα διανύσματα $X_i \in R^2$ των κύκλων και των αστεριών στα παρακάτω Z:

$$Z_1, Z_2, \dots Z_N = K(X, X_1), K(X, X_2), \dots K(X, X_N)$$

Τα οποία είναι ίσα με το πλήθος των διανυσμάτων και ανήκουν στον διανυσματικό χώρο V Έπειτα ορίζουμε γραμμική θήκη $\Omega = \{c_1 Z_1, c_2 Z_2, \dots c_N Z_N\}$. Αν $\Phi(X)$ ανήκει στον διανυσματικό χώρο V ορίζουμε την κάθετη προβολή $\widehat{\Phi}(X) \in \Omega$. Έτσι από την αρχή της ορθογωνιότητας προκύπτει το εξής:

$$< \Phi(X) - \widehat{\Phi}(X), K(X, X_i) > = 0 \Leftrightarrow$$

$$< \Phi(X) K(X, X_i) > = < \widehat{\Phi}(X) K(X, X_i) > \Leftrightarrow$$

$$\Phi(X_i) = \widehat{\Phi}(X_i)$$

Έτσι καταλήγουμε στο ότι μπορούμε σε κάθε X_i να αντικαταστήσουμε την $\Phi(X)$ με την κάθετη προβολή της $\widehat{\Phi}(X)$.

Δηλαδή αν έχουμε:

$$\Phi(X) = \sum_{i=1}^{M} c_i K(X, Z_i), M \leq N$$

Μπορούμε να το αντικαταστήσουμε με το παρακάτω:

$$\widehat{\Phi}(X) = \sum_{X_i \in stars} \alpha_i K(X, X_i) + \sum_{X_j \in stars} \beta_j K(X, X_j)$$

Ερώτημα β)

Χρησιμοποιώντας την αρχή της ορθογωνιότητας θέλουμε να αποδείξουμε ότι:

$$\|\Phi(X)\|^2 = \left\|\widehat{\Phi}(X)\right\|^2 + \left\|\Phi(X) - \widehat{\Phi}(X)\right\|^2 \geq \left\|\widehat{\Phi}(X)\right\|^2$$

Αρχικά ισχύει το εξής:

$$\|\Phi(X)\|^2 = \left\|\widehat{\Phi}(X) + \Phi(X) - \widehat{\Phi}(X)\right\| =$$

$$= \left\|\widehat{\Phi}(X)\right\|^2 + \left\|\Phi(X) - \widehat{\Phi}(X)\right\|^2 + 2 < \widehat{\Phi}(X), \Phi(X) - \widehat{\Phi}(X) >$$

Επιπλέον ισχύει ότι το $\widehat{\Phi}(X)$ είναι κάθετο με το $\Phi(X) - \widehat{\Phi}(X)$ οπότε ισχύει και ότι

$$< \widehat{\Phi}(X), \Phi(X) - \widehat{\Phi}(X) > = 0$$

Έτσι καταλήγουμε στο ζητούμενο, δηλαδή ότι

$$\|\Phi(X)\|^2 = \left\|\hat{\Phi}(X)\right\|^2 + \left\|\Phi(X) - \hat{\Phi}(X)\right\|^2 \geq \left\|\hat{\Phi}(X)\right\|^2$$

Αφού ισχύει η παραπάνω ισότητα καταλαβαίνουμε ότι μπορούμε να αντικαταστήσουμε το $\|\Phi(X)\|^2$ με το $\left\|\hat{\Phi}(X)\right\|^2$ το οποίο είναι μικρότερο και εμείς έχουμε σκοπό να ελαχιστοποιήσουμε.

Ερώτημα γ)

Αφού αποδείξαμε τα παραπάνω θέλουμε να λύσουμε το εξής πρόβλημα βελτιστοποίησης

$$\min_{\hat{\Phi} \in V} \left\{ \sum_{i=1}^{N} \left(Y_i - \hat{\Phi}(X_i)\right)^2 + \lambda \left\|\hat{\Phi}(X)\right\|^2 \right\}$$

$$\Phi(X) = \sum_{i=1}^{N} c_i K(X, X_i) = c_1 K(X, X_1) + \cdots + c_N K(X, X_N)$$

Έστω X το διάνυσμα με τα δεδομένα και Y το διάνυσμα με τις ετικέτες τα οποία φαίνονται παρακάτω:

$$X = [X_1 \quad \dots \quad X_N]^T$$
$$Y = [Y_1 \quad \dots \quad Y_N]^T$$

Έτσι προκύπτει το εξής:

$$\hat{\Phi}(X) = \begin{bmatrix} K(X_1, X_1) & \cdots & K(X_1, X_N) \\ \vdots & \ddots & \vdots \\ K(X_N, X_1) & \cdots & K(X_N, X_N) \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_N \end{bmatrix} = KC$$

δηλαδή:

$$\sum_{i=1}^{N} \left(Y_i - \hat{\Phi}(X_i)\right)^2 = \|Y - KC\|^2$$

Έπειτα βρίσκουμε τις μερικές παραγώγους της σχέσης που θέλουμε να ελαχιστοποιήσουμε ως προς το διάνυσμα C το οποίο και ψάχνουμε

$$\frac{\partial}{\partial C} \sum_{i=1}^{N} \left(Y_i - \hat{\Phi}(X_i)\right)^2 = \frac{\partial}{\partial C} \|Y - KC\|^2 = -2K(Y - KC)$$

Επίσης

$$\frac{\partial}{\partial C} \lambda \left\|\hat{\Phi}(X)\right\|^2 = \lambda * 2KC$$

Από τις παραπάνω 3 σχέσεις βρίσκουμε την εξίσωση του διανύσματος C η οποία φαίνεται παρακάτω:

$$-2K(Y - KC) + \lambda 2KC = 0$$

$$\mathrm{C} = (\mathrm{KC} + \lambda I)^{-1} Y, \text{όπου } I \text{ ο μοναδιαίος πίνακας}$$

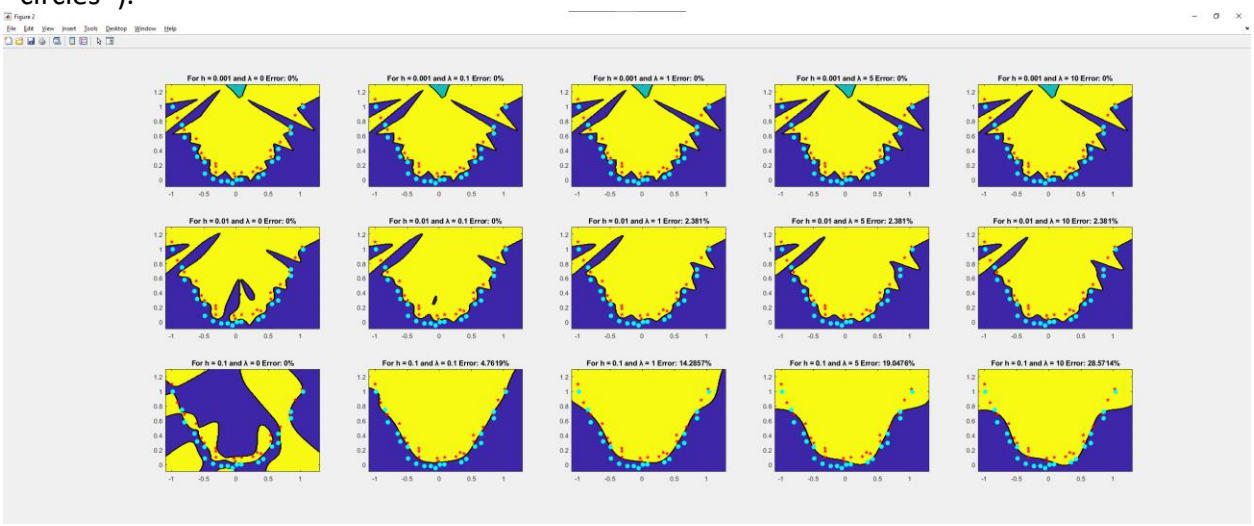
Τέλος αφού στο διάνυσμα X με τα δεδομένα έχουμε πρώτα τα δεδομένα stars και μετά τα δεδομένα circles τα πρώτα 21 c (N_stars = 21) θα είναι τα ζητούμενα α_i ενώ τα επόμενα 21 (N_circles = 21) θα είναι τα ζητούμενα β_i.

Ερώτημα δ)

Για να κατηγοριοποιήσουμε ένα νέο σημείο X_new στην κατηγορία ''star'' ή στην κατηγορία ''circle'' αφού υπολογίσουμε την τιμή $\widehat{\Phi}(X_{new})$ θα την συγκρίνουμε με το 0 και άμα είναι μεγαλύτερη το σημείο θα κατηγοριοποιείται στην κατηγορία ''star'' ενώ άμα είναι μικρότερη το σημείο θα κατηγοριοποιείται στην κατηγορία circle

Ερώτημα ε)

Για να σχεδιαστεί το διαχωριστικό σύνορο πηγαίνει σε σημεία σε όλο τον χώρο και τα κατηγοριοποιούμε δηλαδή πάλι όπου η τιμή που παίρνουμε είναι μεγαλύτερη του 0 αυτό το σημείο κατηγοριοποιείται στην κατηγορία ''stars'' (label = 1) ενώ όπου παίρνουμε τιμή μικρότερη του μηδενός το σημείο κατηγοριοποιείται στην κατηγορία ''circles''. Παρακάτω βλέπουμε τα διαχωριστικά σύνορα που υπολογίστηκαν για διάφορες τιμές του h (0.001, 0.01, 0.1) και του λ (0, 0.1, 1, 5, 10). Η κίτρινη περιοχή σε κάθε φωτογραφία είναι η περιοχή των αστεριών ενώ η μωβ είναι η περιοχή των κύκλων. Επίσης σε κάθε φωτογραφία φαίνονται και τα δεδομένα εκπαίδευσης που δόθηκαν (με κόκκινο αστέρι τα ''stars'' ενώ με γαλάζιο κύκλο τα ''circles'').

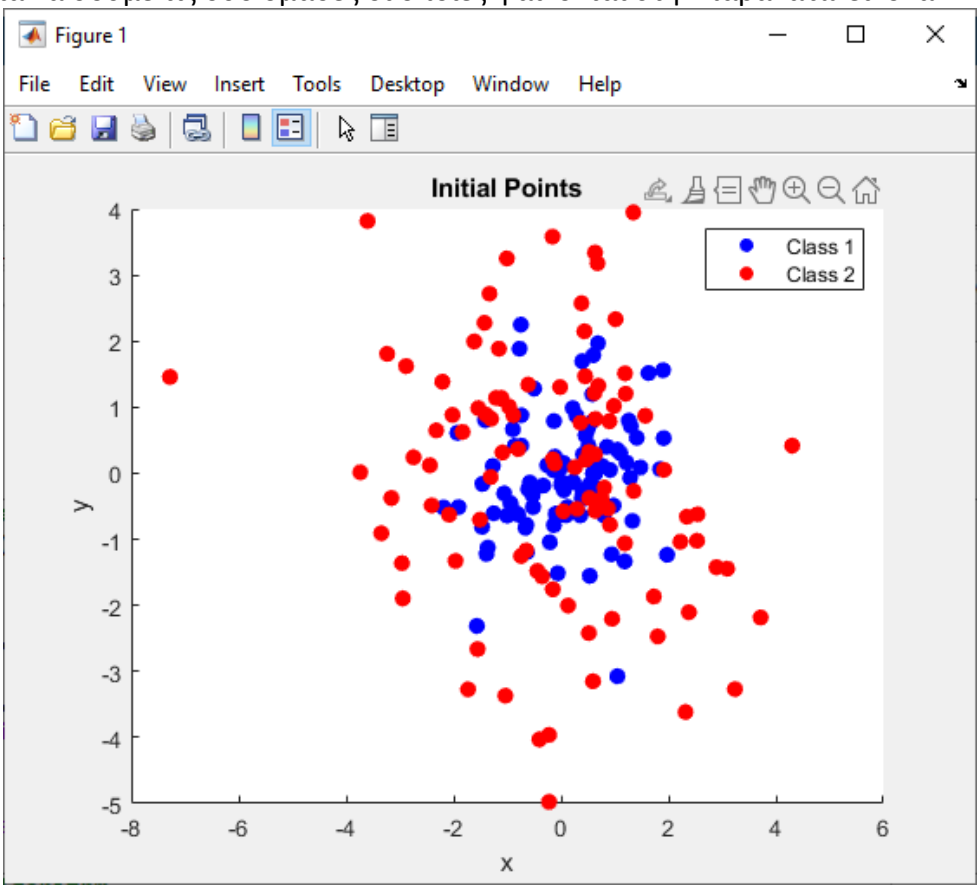


Το πιο καλό σύνορο φαίνεται να είναι αυτό της $2^{ης}$ εικόνας της τελευταίας γραμμής από τα αριστερά για h = 0.1 και λ = 0.1. Τα σφάλματα για τα αρχικά δεδομένα για κάθε συνδυασμό φαίνονται στον παρακάτω πίνακα:

h\λ	**0**	**0.1**	**1**	**5**	**10**
0.001	0%	0%	0%	0%	0%
0.01	0%	0%	2.38%	2.38%	2.38%
0.1	0%	4.76%	14.29%	19.05%	28.57%

Πρόβλημα 3.3

Στο συγκεκριμένο πρόβλημα μας δίνονται 200 διανύσματα μήκους 2 τα οποία θέλουμε να τα ομαδοποιήσουμε σε δύο ομάδες. Επιπλέον μας δίνεται η μυστική πληροφορία ότι τα πρώτα 100 διανύσματα ανήκουν στην μια ομάδα ενώ τα 100 επόμενα ανήκουν στην άλλη. Έτσι χρησιμοποιώντας αυτή την μυστική πληροφορία χρησιμοποιούμε την εντολή scatter() της MATLAB για να δούμε τις δύο ομάδες οι οποίες φαίνονται στην παρακάτω εικόνα:



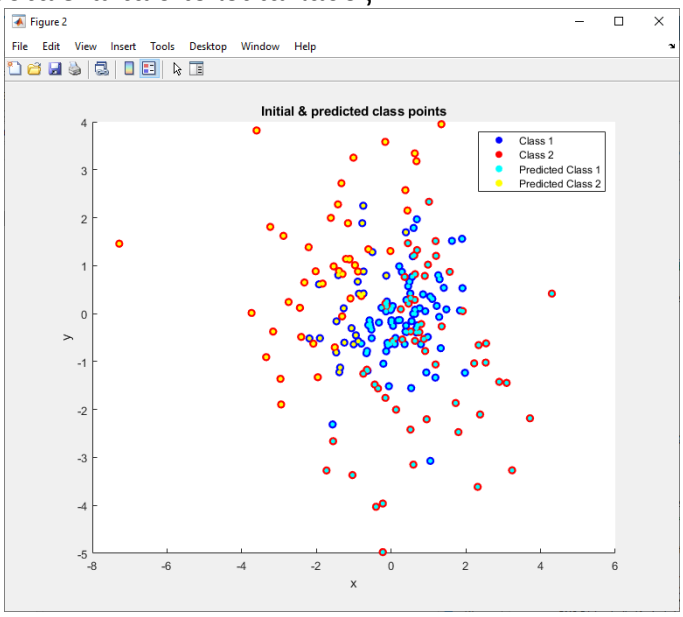
Ερώτημα α)

Θέλουμε χρησιμοποιώντας τον αλγόριθμο K-Means να προσπαθήσουμε να ομαδοποιήσουμε τα παραπάνω σημεία σε δύο ομάδες. Ο αλγόριθμος ακολουθεί τα παρακάτω βήματα:

1. Αρχικά επιλέγουμε K (αριθμός ομάδων) αντιπροσώπους από τα παραδείγματα $Z_1,\ldots Z_K$
2. Δημιουργούμε ομάδες $C_1,\ldots C_K$. Η κάθε ομάδα C_n αποτελείται από τα παραδείγματα που απέχουν λιγότερο από το αντίστοιχο κέντρο Z_n
3. Βρίσκουμε τα νέα κέντρα κάθε ομάδας υπολογίζοντας τον μέσο όρο τον παραδειγμάτων της
4. Επαναλαμβάνουμε την διαδικασία από το βήμα 2 και έπειτα μέχρι η συνολική διασπορά να συγκλίνει σε κάποια σταθερή τιμή.

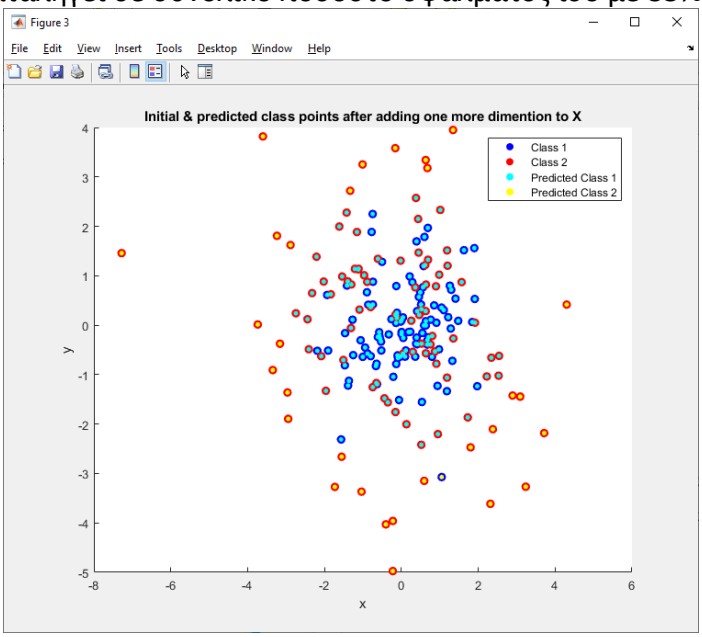
Έπειτα για να βρούμε το error του classification ελέγχουμε σε ποιο από τα δύο τελικά κέντρα που υπολογίσαμε είναι πιο κοντά τα σημεία που μας δίνονται και τα εντάσσουμε σε μία κλάση και συγκεκριμένα θεωρούμε ότι τα σημεία που είναι πιο κοντά στο πρώτο κέντρο είναι τα δεδομένα της πρώτης κλάσης ενώ αυτά που είναι πιο κοντά στο δεύτερο της δεύτερης.

Καθώς τελειώσει ο αλγόριθμος το συνολικό ποσοστό σφάλματος που παίρνουμε είναι 39%. Στην παρακάτω εικόνα φαίνονται τα αρχικά σημεία η κλάσεις στις οποίες έχουν ταξινομηθεί από τον αλγόριθμο. Τα σημεία που έχουν ίδιο χρώμα στο εσωτερικό με το εξωτερικό τους έχουν ταξινομηθεί σωστά ενώ τα υπόλοιπα λάθος.



Ερώτημα β)

Σκοπός μας είναι να βελτιστοποιήσουμε την απόδοση του αλγόριθμου K-means. Συγκεκριμένα θέλουμε να αντιμετωπίσουμε την τάση του αλγόριθμου να δημιουργεί γραμμικά διαχωριστικά όρια. Για να το πετύχουμε αυτό αντικαταστούμε κάθε δισδιάστατο X_i εισάγοντας μια τρίτη συντεταγμένη. Συγκεκριμένα το αντικαταστούμε με το τρισδιάστατο $\{X_i, \|X_i\|^2\}$.
Έτσι ο αλγόριθμος καταλήγει σε συνολικό ποσοστό σφάλματος ίσο με 35%.



Όπως βλέπουμε ό αλγόριθμος κάνει λάθος μόνο σε ένα από τα μπλε σημεία και τα ταξινομεί σωστά τα κόκκινα σημεία τα οποία βρίσκονται μακριά από την αρχή των αξόνων.

Κώδικες

Κώδικας Προβλήματος 3.1

```
%% Task 3.1
clc;
clear;
close all;

%% Aproximate the probability dense with the Kernel
syms var;
rect = rectangularPulse(0,1,var); % Create a rectangular pulse for ploting

sz = [1 1000]; % Set the amount of implementations
samples = 4 * sz(2); % Set the number of the samples for x axis

x = rand(sz); % Generate random implementations in range [0, 1]
x_axis = linspace(-3, 4, samples); % The x axis values
x = x_axis' - x; % Calculate the value for the kernel

hi = [0.0001 0.001 0.01 0.1];

for i=1:size(hi,2)
    h = hi(i);
    f_hat = mean(gaussianKernel(x,h), 2); % Aproximate the probability dense with
the Kernel
    %==========PLOT==========%
    subplot(2,2,i)
    plot(x_axis, f_hat)
    hold on;
    fplot(rect, 'r--')
    title(['h = ' num2str(h)])
    axis([-3 4 0 1.5])
    %========================%
end

%% Functions
function prob_dens = gaussianKernel(x,h) % Gaussian Kernel
    prob_dens = exp(-0.5 * (x.^2) / h) / (sqrt(2*pi*h)) ;
end
```

Κώδικας Προβλήματος 3.2

```
%% Task 3.2
clc;
clear;
close all;

%% Load and plot given data
data = load("data32.mat");
stars = data.stars; % Get star points
circles = data.circles; % Get circle points
%==========PLOT==========%
figure()
scatter(stars(:,1), stars(:,2), 50, 'rp', 'filled')
hold on;
scatter(circles(:,1), circles(:,2), 50, 'cyan', 'filled')
title('Star & Circle Points')
xlabel('x1')
ylabel('x2')
xlim([-1.1 1.3])
ylim([-0.1 1.3])
legend('Stars', 'Circles')
%========================%

%% Calculate Errors and plot each class area
h = [0.001 0.01 0.1]; % Set h values
l = [0 0.1 1 5 10]; % Set l values

stars_n = size(stars, 1); % Stars amount
circles_n = size(circles,1); % Circles amount

star_labels = ones([1, stars_n]); % Label Stars as 1
circle_labels = -ones([1, circles_n]); % Label Circles as -1

X = [stars; circles]; % Stack stars and cricles
Y = [star_labels circle_labels]'; % Stack the labels

X = reshape(X' , [1, 2, stars_n+circles_n]) - X; % Calculate the value for the
Kernel

plot_cnt = 1;
figure()

for i=1:size(h,2)
    disp('%===============%')
    disp(['For h = ' num2str(h(i))])
    disp('%---------------%')
```

```
        for j=1:size(l,2)
                disp(['Calculating error for l = ' num2str(l(j))])

                % Calculate parameter vector
                C = (inv(K(X, h(i), stars_n+circles_n) + l(j) * eye(stars_n +
        circles_n))) * Y;

                a = C(1:stars_n)'; % Take the first 21 parametes that are for the stars
                b = C(circles_n+1:end)'; % Take the last 21 parameters that are for the
        circels

                star_error = calculateError(stars, stars, circles, a, b, h(i));
                star_error = star_error(2); % Calcualte the star error

                circle_error = calculateError(circles, stars, circles, a, b, h(i));
                circle_error = circle_error(1); % Calculate the circle error

                total_error = (star_error + circle_error)/(stars_n +circles_n); %
        Calculate total error
                disp(['Total error: ' num2str(total_error*100) '%'])

                lin_x = linspace(-1.1, 1.3, 200);
                lin_y = linspace(-0.1, 1.3, 200);

                [x_mesh, y_mesh] = meshgrid(lin_x, lin_y);

                x = reshape(x_mesh', [size(x_mesh,1)^2, 1]);
                y = reshape(y_mesh', [size(y_mesh,1)^2, 1]);

                x = [x y];
                n = size(x,1);

                % Calculate the values for the Kernel
                X_a = reshape(x', [1,2,n]) - stars;
                X_b = reshape(x', [1,2,n]) - circles;

                t = sum(a * K(X_a, h(i), n), 1) +  sum(b * K(X_b, h(i), n), 1);

                srf = reshape(t,[length(lin_x),length(lin_y)])'; % Reshape the surface
        for ploting

                srf(srf>0) = 1; % Where the surface is positive it's the stars area
                srf(srf<0) =-1; % Where the surface is negative it's the circles area
          %==========PLOT==========%
          subplot(size(h,2), size(l,2), plot_cnt)
          contourf(x_mesh,y_mesh,srf);
          hold on;
          scatter(stars(:,1), stars(:,2), 50, 'rp', 'filled')
          hold on;
          scatter(circles(:,1), circles(:,2), 50, 'cyan', 'filled')
          title(['For h = ' num2str(h(i)) ' and λ = ' num2str(l(j)) ' Error: '
num2str(total_error*100) '%']);
          %========================%
          plot_cnt = plot_cnt + 1;
    end
    disp('%===============%')
end
```

```
%% Functions
function prob_dens = K(x, h , n) % Kernel
    nrm = vecnorm(x,2,2);
    prob_dens  = exp((-1/h)*(reshape(nrm,[size(x,1),n]).^2));
end

function error = calculateError(x, stars, circles, a, b, h)
    error = [0 0];
    n = size(x,1);

    X_a = reshape(x', [1,2,n]) - stars;
    X_b = reshape(x', [1,2,n]) - circles;

    t = sum(a * K(X_a, h, n), 1) +  sum(b * K(X_b, h, n),1);

    error(1) = nnz(t>0);
    error(2) = nnz(t<0);
end
```

Κώδικας Προβλήματος 3.3

```
%% Task 3.3
clc;
clear;
close all;

%% Load Data
data = load("data33.mat");
X = data.X;
figure()
scatter(X(1,1:100), X(2,1:100), 50, 'blue', 'filled')
hold on;
scatter(X(1,101:end), X(2,101:end), 50, 'red', 'filled')
title('Initial Points')
xlabel('x')
ylabel('y')
legend('Class 1', 'Class 2')

%% K-means Algorithm
K = 2; % Set the amount of Z's
iter = 25; % Set the amount of iterations
[C, Z] = KMeans(X,K,iter); % Run the K-means algorithm
Class_Err = classificationError(X,Z); % Calculate the classification error
disp(['Total classification error: ' num2str(Class_Err * 100) '%'])

%% Optimize K-means
X_Norm = vecnorm(X,2,1);
X_New = [X ; X_Norm.^2];
[C_b, Z_b] = KMeans(X_New,K,iter); % Run the K-means algorithm
Class_Err_Opt = classificationError(X_New,Z_b); % Calculate the classification
error
disp(['Total classification error: ' num2str(Class_Err_Opt * 100) '%'])
```

```
%% Plot Results and Data
Plot_points = cell(1,K);
Plot_points_opt = cell(1,K);

for i=1:size(X,1)
    Class_ind = C{i};
    Class_ind_opt = C_b{i};
    Class_points = X(:, Class_ind);
    Class_points_opt = X(:,Class_ind_opt);
    Plot_points{i} = Class_points;
    Plot_points_opt{i} = Class_points_opt;
end

%==========PLOTS==========%
figure()
scatter(X(1,1:100), X(2,1:100), 50, 'blue', 'filled')
hold on;
scatter(X(1,101:end), X(2,101:end), 50, 'red', 'filled')
hold on;
scatter(Plot_points{1}(1,:), Plot_points{1}(2,:), 15, 'cyan', 'filled')
hold on;
scatter(Plot_points{2}(1,:), Plot_points{2}(2,:), 15, 'yellow', 'filled')
title('Initial & predicted class points')
xlabel('x')
ylabel('y')
legend('Class 1','Class 2', 'Predicted Class 1', 'Predicted Class 2')

%==========PLOTS==========%
figure()
scatter(X(1,1:100), X(2,1:100), 50, 'blue', 'filled')
hold on;
scatter(X(1,101:end), X(2,101:end), 50, 'red', 'filled')
hold on;
scatter(Plot_points_opt{1}(1,:), Plot_points_opt{1}(2,:), 15, 'cyan', 'filled')
hold on;
scatter(Plot_points_opt{2}(1,:), Plot_points_opt{2}(2,:), 15, 'yellow', 'filled')
title('Initial & predicted class points after adding one more dimention to X')
xlabel('x')
ylabel('y')
legend('Class 1','Class 2', 'Predicted Class 1', 'Predicted Class 2')

%% Functions
function err = classificationError(X,Z)
    tot_err_1 = 0;
    tot_err_2 = 0;
    maxi = size(X,2);

    for i=1:maxi
        [~ , index] = min(vecnorm(Z'-X(:,i)',2,2));
        if index == 1 && i >=100
            tot_err_1 = tot_err_1 + 1;

        elseif index == 2 && i<100
            tot_err_2 = tot_err_2 + 1;
        end
    end
    err = (tot_err_1 + tot_err_2)/maxi;
end
```

```
function [C, Z] = KMeans(X, K, iterations)
    % Choose randomly Z
    sz = [size(X,1), K];
    Z = zeros(sz);
    maxi = size(X,2);
    all_distances = zeros([1,iterations]);

    for i=1:K % Initialize Centers
        center_ind = randi(maxi);
        center = X(:,center_ind);
        Z(:,i) = center ;
    end

    for iter=1:iterations % Run the K-means algorithm
        distance = 0;
        C = cell(1, K);

        for i=1:K % Initialize C
            C{i} = [];
        end

        for i=1:maxi % Classify data
            [~ , index] = min(vecnorm(Z'-X(:,i)',2,2));
            C{index}(end + 1) = i;
        end

        for i=1:K % Change the centers
            class_indecies = C{i};
            class_points = X(:, class_indecies);
            distance = distance + sum(vecnorm(Z(:,i)' - class_points', 2, 2));
            Z(:,i) = mean(class_points,2);
        end
        all_distances(iter) = distance;
    end
    disp('%======K-means Result======%')
end
```