**Μηχανική Μάθηση**

**3o Σετ Ασκήσεων**

ΟΝΟΜΑ: Βασίλειος

ΕΠΩΝΥΜΟ: Καλαϊτζόπουλος

ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ: 1066670

ΕΤΟΣ ΦΟΙΤΗΣΗΣ: 4o

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:

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

όπου xn 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:

Graphical user interface, chart, box and whisker chart

Description automatically generated

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(Χ) όπου Χ = [x1, x2]T τον οποίο όταν τον εφαρμόζουμε στα σημεία θα το αντιστοιχεί στην σωστή ‘’ετικέτα’’. Για να βρούμε τον μετασχηματισμό Φ(Χ) λύνουμε το παρακάτω πρόβλημα βελτιστοποίησης:

όπου Ν 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:

Τα δεδομένα τα οποία θέλουμε να διαχωρίσουμε φαίνονται παρακάτω:

Chart, scatter chart

Description automatically generated

## Ερώτημα α)

Θέλουμε να αποδείξουμε μέσω του Representer Theorem ότι μπορούμε στα πρώτα δύο αθροίσματα να αντικαταστήσουμε το με την ορθογώνια προβολή του πάνω στον γραμμικό υποχώρο που δημιουργούν οι συναρτήσεις όπου και η όπου δηλαδή:

Αρχικά αντιστοιχίζουμε τα διανύσματα των κύκλων και των αστεριών στα παρακάτω Ζ:

Τα οποία είναι ίσα με το πλήθος των διανυσμάτων και ανήκουν στον διανυσματικό χώρο V

Έπειτα ορίζουμε γραμμική θήκη . Αν ανήκει στον διανυσματικό χώρο V ορίζουμε την κάθετη προβολή . Έτσι από την αρχή της ορθογωνιότητας προκύπτει το εξής:

Έτσι καταλήγουμε στο ότι μπορούμε σε κάθε να αντικαταστήσουμε την με την κάθετη προβολή της .

Δηλαδή αν έχουμε:

Μπορούμε να το αντικαταστήσουμε με το παρακάτω:

## Ερώτημα β)

Χρησιμοποιώντας την αρχή της ορθογωνιότητας θέλουμε να αποδείξουμε ότι:

Αρχικά ισχύει το εξής:

Επιπλέον ισχύει ότι το είναι κάθετο με το οπότε ισχύει και ότι

Έτσι καταλήγουμε στο ζητούμενο, δηλαδή ότι

Αφού ισχύει η παραπάνω ισότητα καταλαβαίνουμε ότι μπορούμε να αντικαταστήσουμε το με το το οποίο είναι μικρότερο και εμείς έχουμε σκοπό να ελαχιστοποιήσουμε.

## Ερώτημα γ)

Αφού αποδείξαμε τα παραπάνω θέλουμε να λύσουμε το εξής πρόβλημα βελτιστοποίησης

Έστω Χ το διάνυσμα με τα δεδομένα και Υ το διάνυσμα με τις ετικέτες τα οποία φαίνονται παρακάτω:

Έτσι προκύπτει το εξής:

δηλαδή:

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

Επίσης

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

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 αi ενώ τα επόμενα 21 (N\_circles = 21) θα είναι τα ζητούμενα βi.

## Ερώτημα δ)

Για να κατηγοριοποιήσουμε ένα νέο σημείο X\_new στην κατηγορία ‘’star’’ ή στην κατηγορία ‘’circle’’ αφού υπολογίσουμε την τιμή 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). H 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’’).

Shape

Description automatically generated

Το πιο καλό σύνορο φαίνεται να είναι αυτό της 2ης 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:

Chart, scatter chart

Description automatically generated

## 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 Κ (αριθμός ομάδων) αντιπροσώπους από τα παραδείγματα Ζ1,…Ζκ
2. Δημιουργούμε ομάδες C1,…CK. Η κάθε ομάδα Cn αποτελείται από τα παραδείγματα που απέχουν λιγότερο από το αντίστοιχο κέντρο Zn
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.

Chart, scatter chart

Description automatically generated

## 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 εισάγοντας μια τρίτη συντεταγμένη. Συγκεκριμένα το αντικαταστούμε με το τρισδιάστατο .

Έτσι ο αλγόριθμος καταλήγει σε συνολικό ποσοστό σφάλματος ίσο με 35%.

Chart, scatter chart

Description automatically generated

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

## 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) '%'])

%% 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

%% 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