Προβλεπτική Αναλυτική

Μεταπτυχιακή απαλλακτική εργασία ΙΟΥΝΙΟΥ 2024

Καθηγητής:
Μ. Φιλιππάκης

Μαρκουλής Δημήτριος
Καμπάσης Παναγιώτης
Πανεπιστήμιο Πειραιώς
Τμήμα Ψηφιακών Συστημάτων
Μεγάλα Δεδομένα και Αναλυτική
25 Ιουνίου 2024

Προβλεπτική Αναλυτική

Πάνος Καμπάσης
27/05/2024

Περιεχόμενα

I *Linear Regression* 2
- I-A′ *Simple linear Regression* . . . 2
- I-B′ *Linear Regression* με *Stochastic Gradient Descent* (*SGD*) . . . 3

II *Multivariate Linear Regression* 4
- II-A′ *Multivariate Linear Regression* με *Stochastic Gradient Descent* . . . 4

III *Classification* 5
- III-A′ *Logistic Regression* . . . 5
- III-B′ *Logistic Regression* με την μέθοδο *Newton* . . . 6

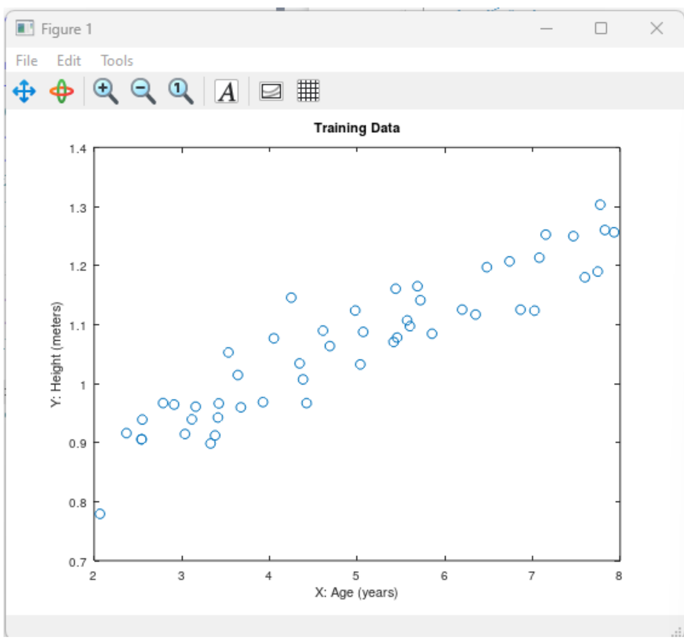
IV *Unsupervised Learning* 7
- IV-A′ $K - meansClustering$ . . . 7

I. *Linear Regression*

Task 1

A′. *Simple linear Regression*

Χρησιμοποιώντας το κατάλληλο στατιστικό πακέτο της *Octave* υλοποιούμε το μοντέλο της απλής γραμμικής παλινδρόμησης για περιβάλλοντα τύπου *Matlab* και εξετάζουμε τα αποτελέσματα, τους χρόνους σύγκλισης και τα ποιοτικά χαρακτηριστικά της εκάστοτε μεθόδου σε συνδυασμό με το ποιόν των αρχικών μας δεδομένων. Οι μέθοδοι αυτοί έχουν ενίοτε την δυνατότητα να προσδώσουν προστιθέμενη πληροφορία στα δεδομένα μας και να εκμαιεύσουν χαρακτηριστικά που πρότινος δεν ήταν διαυγή.
Φορτώνουμε το στατιστικό πακέτο της *Octave* με: *pkgload statistics* (που είναι η αντίστοιχη βιβλιοθήκη της *MATLAB's Statistics and Machine Learning Toolbox*).
Για δεδομένα x, y:



Χρησιμοποιώντας τον παρακάτω κώδικα και εκτέλωντας τον στην *Octave* παίρνουμε τα εξής αποτελέσματα.
Επομένως το αποτέλεσμα μας είναι: *Theta values*:

$$\hat{\theta}_1 = 0.750163$$
$$\hat{\theta}_2 = 0.063881$$

Επομένως η ευθεία μας προσαρμόζεται ως:

$$y = \theta_1 + \theta_2 x + \epsilon \quad (1)$$

```
clear all; close all; clc;
x = load('linear_regressionx.dat');
y = load('linear_regressiony.dat');

m = length(y); % number of data points = 50

% Add a column of ones to x for the intercept term
X = [ones(m, 1), x];

% Fit the linear regression model using regress
theta = regress(y, X);

% Predict y values using the model
y_pred = X * theta;

% Visualize the training data and the regression line
figure;
plot(x, y, 'o'); % Plot the data points
hold on;
plot(x, y_pred, '-r'); % Plot the regression line
ylabel('Y: Height (meters)')
xlabel('X: Age (years)')
title('Simple Linear Regression using regress(x, y)')
legend('Training data', 'Linear regression')
hold off;

% Display the theta values
disp('Using regress function:');
disp('Coefficients (Intercept and Slope):');
disp(theta);
```

where :

y : *Dependent variable (response)*
x : *Independent variable (predictor)*
θ_1 : *Intercept*
θ_2 : *Slope of the regression line*
ϵ : *Error term*

Estimated Regression Equation

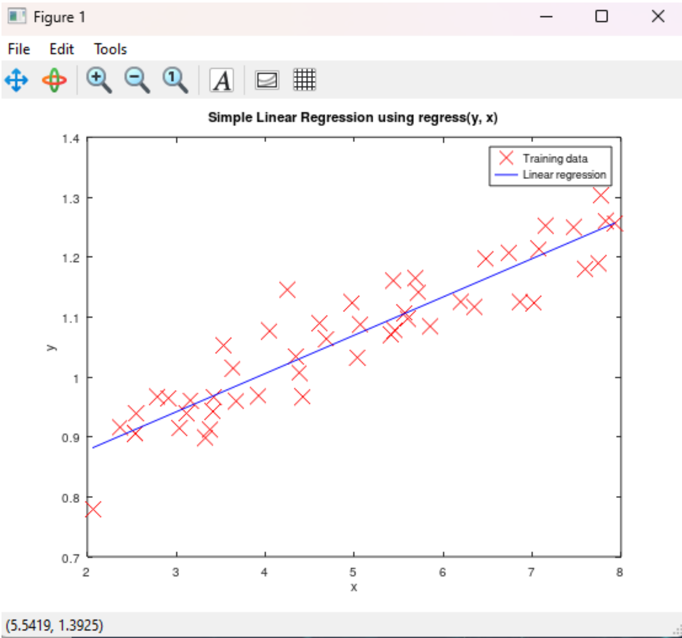
The estimated regression equation is given by:

$$\hat{y} = \hat{\theta}_1 + \hat{\theta}_2 x \quad (2)$$

where:

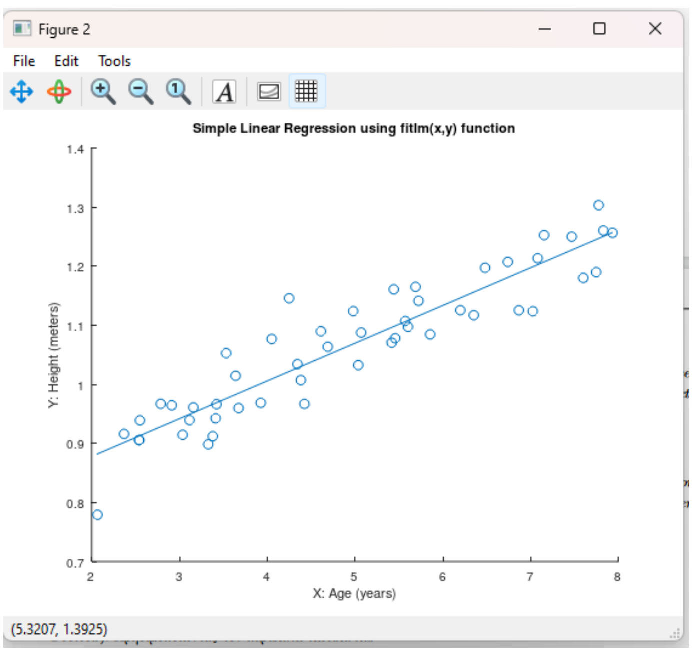
$\hat{y}$: *Predicted value of y*
$\hat{\theta}_0$: *Estimated intercept*
$\hat{\theta}_1$: *Estimated slope*

Και ο αντίστοιχος γράφος:



Σχήμα 1: Γράφημα ΑΓΠ.

Συγκρίνουμε το αποτέλεσμα μας με εκείνο του αρχείου $simple\_lineaerregression.m$ και παρατηρούμε:



Σχήμα 2: Γράφημα ΑΓΠ.

$$\hat{\beta}_0 = 0.750163$$
$$\hat{\beta}_1 = 0.063881$$

Βλέπουμε ότι τα αποτελέσματα είναι ίδια καθώς έχουμε ακριβώς τις ίδιες εκτιμήσεις ϑ. Η μια προέκυψε από την $regress(y, X)$ ενώ η άλλη από $fitlm(x, y)$

Task 2

B′. *Linear Regression* με *Stochastic Gradient Descent (SGD)*

Ομοίως ανακαλούμε την παραπάνω υλοποίηση προσθέτοντας μια άλλη μέθοδο σύγκλισης για την εύρεση της καλύτερης δυνατής κλίσης. Η συνάρτηση μας είναι:

$$h_\theta(x) = \theta^T x = \sum_{i=0}^{n} \theta_i x_i$$

Και με κανόνα ανενέωσης για να βρούμε το κατάλληλο ϑ:

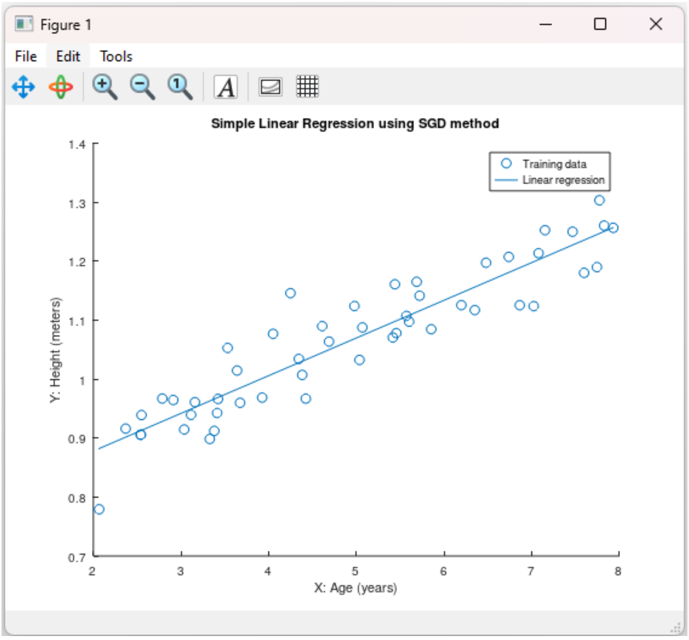
$$\theta_i := \theta_i - \alpha \frac{1}{m} \sum_{t=1}^{m} \left(h_\theta(x^{(t)}) - y^{(t)} \right) x_j^{(t)}$$

Χρησιμοποιούμε τον κώδικα $Task1.m$:

```
alpha = 0.07; % Learning rate
iterations = 1500; % Number of iterations
theta = zeros(2, 1); % Initialize fitting parameters

% Add a column of ones to x for the intercept term
X = [ones(m, 1), x];

for iter = 1:iterations
    gradient = (1/m) * X' * (X * theta - y); % Compute the gradient
    theta = theta - alpha * gradient; % Update theta
end

% Visualize results
figure;
hold on;
scatter(x, y);
plot(x, X * theta, '-');
ylabel('Y: Height (meters)')
xlabel('X: Age (years)')
title({'Simple Linear Regression using SGD method'});
legend('Training data', 'Linear regression');
hold off;

% Display theta values
disp('Using Stochastic Gradient Descent:');
disp('Coefficients (Intercept and Slope):');
disp(theta);
```

Σχήμα 3: *Simple linear Regression with Stochastic Gradient Descent*

Έχουμε το παρακάτω γράφημα που είναι σχεδόν πανομοιότυπο με τις προηγούμενες μεθόδους: με ϑ:



Σχήμα 4: *Simple linear Regression with Stochastic Gradient Descent*

$$\begin{bmatrix} 0.750150 \\ 0.063883 \end{bmatrix}$$

II. *Multivariate Linear Regression*

*Task*3

Α'. *Multivariate Linear Regression* με *Stochastic Gradient Descent*

Χρησιμοποιώντας νέα δεδομένα μεγαλύτερων διαστάσεων και τον κώδικα *Matlab* *multivariate_linear_regression.m* Υλοποιούμε την παραπάνω διαδικασία για μια ακόμα φορά σε μεγαλύτερες διαστάσεις (γεγονός που βοηθά την εκμαίευση περισσότερης πληροφορίας καθώς στις πολλαπλές διαστάσεις εύκολα χάνεται η αντιληπτική ικανότητα) και ρυθμίζουμε τις διάφορες τιμές του α.
Θα χρησιμοποιήσουμε το παρακάτω εύρος τιμών για να ερευνήσουμε την συμπεριφορά της σύγκλισης

$$alpha = [0.01, 0.03, 0.1, 0.3, 1, 1.3]$$

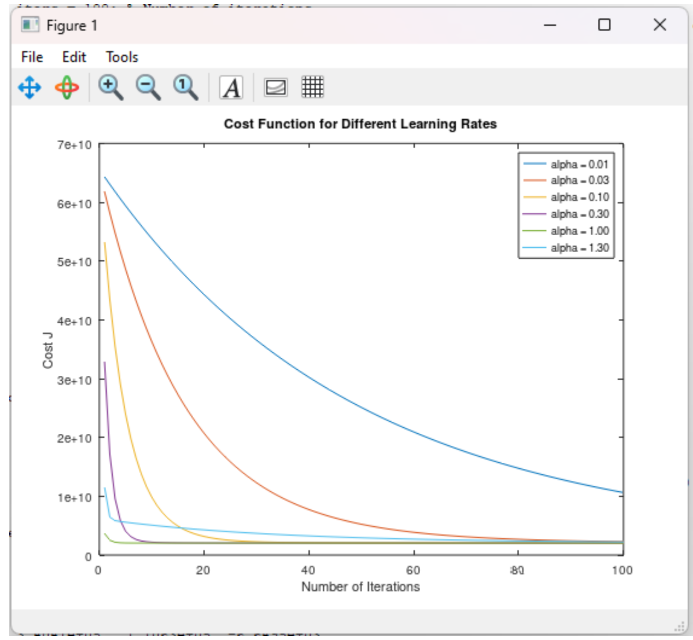
Τότε ο κώδικας *Task*3*.m* μας δίνει:
Καταγράφουμε επίσης και τις παρακάτω τιμές για τις διάφορες περιπτώσεις:

Σύγκριση Ρυθμού Μάθησης και Ανάλυση Σύγκλισης

Άλφα $= 0.01$

- Διαφορά: $[-1.2460 \times 10^5, -4.9247 \times 10^4, 2.6923 \times 10^4]$

Οι διαφορές είναι αρκετά μεγάλες, υποδηλώνοντας ότι ο ρυθμός μάθησης είναι πολύ μικρός και ο αλγόριθμος δεν έχει συγκλίνει καλά στις βέλτιστες τιμές ϑήτα.



Σχήμα 5: Γράφημα σύγκλισης το σ

```
Command Window
Theta values for different alphas:
   2.1581e+05   6.1384e+04   2.0274e+04
   3.2423e+05   9.4178e+04   8.8467e+03
   3.4040e+05   1.0991e+05  -5.9311e+03
   3.4041e+05   1.1063e+05  -6.6494e+03
   3.4041e+05   1.1063e+05  -6.6495e+03
   3.4041e+05   9.9381e+04  -1.7899e+04
Theta values from the normal equation:
   3.4041e+05
   1.1063e+05
  -6.6495e+03
Difference from normal equation theta for different alphas:
  -1.2460e+05  -4.9247e+04   2.6923e+04
  -1.6187e+04  -1.6453e+04   1.5496e+04
  -9.0418e+00  -7.1837e+02   7.1837e+02
  -1.1642e-10  -5.7542e-02   5.7542e-02
            0   4.3656e-11  -3.9108e-11
            0  -1.1250e+04  -1.1250e+04
>> |
```

Σχήμα 6: *Simple linear Regression with Stochastic Gradient Descent*

Άλφα $= 0.03$

- Διαφορά: $[-1.6187 \times 10^4, -1.6453 \times 10^4, 1.5496 \times 10^4]$

Οι διαφορές είναι μικρότερες από ότι για $\alpha = 0.01$, αλλά ακόμα σημαντικές, υποδηλώνοντας καλύτερη σύγκλιση αλλά ακόμα όχι βέλτιστη.

Άλφα $= 0.1$

- Διαφορά: $[-9.0418, -7.1837 \times 10^2, 7.1837 \times 10^2]$

Οι διαφορές είναι πολύ μικρότερες, υποδηλώνοντας ότι ο ρυθμός μάθησης είναι πιο κοντά στο βέλτιστο και ο αλγόριθμος έχει σχεδόν συγκλίνει στις σωστές τιμές ϑήτα.

Άλφα $= 0.3$

- Διαφορά: $[-1.1642 \times 10^{-10}, -5.7542 \times 10^{-2}, 5.7542 \times 10^{-2}]$

Οι διαφορές είναι εξαιρετικά μικρές, υποδηλώνοντας εξαιρετική σύγκλιση και σχεδόν τέλεια συμφωνία με τις τιμές θήτα της κανονικής εξίσωσης.

Άλφα $= 1.0$

- Διαφορά: $[0, 4.3656 \times 10^{-11}, -3.9108 \times 10^{-11}]$

Οι διαφορές είναι ουσιαστικά μηδενικές, υποδηλώνοντας τέλεια σύγκλιση στις τιμές θήτα της κανονικής εξίσωσης.

Άλφα $= 1.3$

- Διαφορά: $[0, -1.1250 \times 10^{4}, -1.1250 \times 10^{4}]$

Οι διαφορές είναι πάλι μεγαλύτερες, υποδηλώνοντας ότι ο ρυθμός μάθησης είναι πολύ υψηλός και προκαλεί απόκλιση ή υπέρβαση.
Από την παραπάνω ανάλυση και το γράφημα που μας υποδεικνύει τα 2 καλύτερα άλφα με γαλάζιο και πράσινο καταλήγουμε στο

Συμπέρασμα

- Καλύτερο Άλφα: $\alpha = 1.0$
- Λόγος: Για $\alpha = 1.0$, οι διαφορές μεταξύ των τιμών θήτα του *SGD* και των τιμών θήτα της κανονικής εξίσωσης είναι ουσιαστικά μηδενικές. Αυτό υποδηλώνει τέλεια σύγκλιση και ότι ο ρυθμός μάθησης είναι βέλτιστος για αυτό το συγκεκριμένο πρόβλημα.

III. *Classification*

Task 4

A′. *Logistic Regression*

Χρησιμοποιώντας νέα αριθμητικά δεδομένα με σκοπό την μελέτη σύγκλισης με την σιγμοειδή συνάρτηση έχουμε τις δύο κλάσεις μαθητών που πέτυχαν την είσοδο τους στο πανεπιστήμιο και εκείνων που δεν τα κατάφεραν εκείνη την χρονιά. Κατασκευάζουμε το αρχείο $logistic_regression.m$ και χρησιμοποιώντας την παρακάτω μορφή και *SGD* όπως παραπάνω, έχουμε:

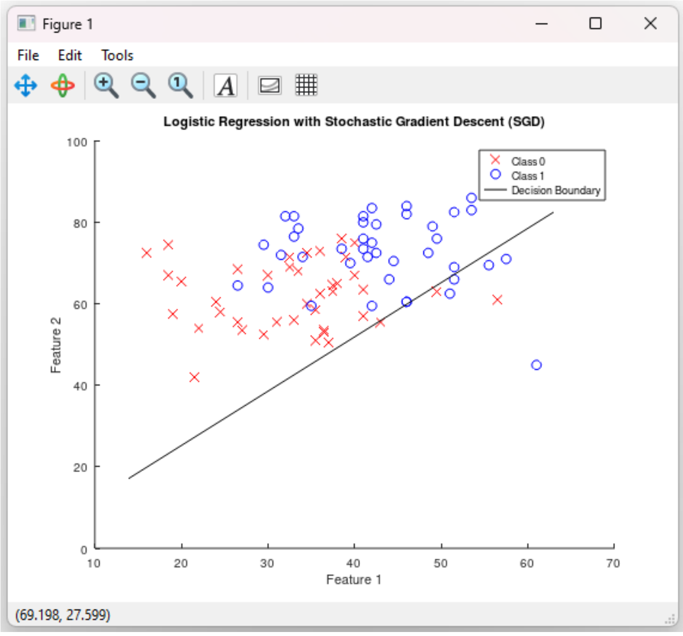
$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}} = P(y = 1|x; \theta)$$

,και:

```
% Logistic Regression with Stochastic Gradient Descent (SGD)
% We will use log_regressionx.dat and log_regressiony.dat
% x = features
% y = labels (0 or 1)
pkg load statistics
clear all; close all; clc;

% Load data
x = load('log_regressionx.dat');
y = load('log_regressiony.dat');

% Initialize parameters
[m, n] = size(x);
x = [ones(m, 1), x]; % Add a column of ones to x for the intercept term
theta = zeros(n + 1, 1); % Initialize fitting parameters

alpha = 0.01; % Learning rate
iterations = 1500; % Number of iterations

% Sigmoid function
sigmoid = @(z) 1 ./ (1 + exp(-z));

% Gradient Descent
for iter = 1:iterations
    % Compute hypothesis
    h = sigmoid(x * theta);
    % Compute gradient
    gradient = (1/m) * x' * (h - y);
    % Update theta
    theta = theta - alpha * gradient;
end

% Visualize results
figure;
gscatter(x(:, 2), x(:, 3), y, 'rb', 'xo');
hold on;
x_values = [min(x(:, 2)) - 2, max(x(:, 2)) + 2];
y_values = -(theta(1) + theta(2) * x_values) / theta(3);
plot(x_values, y_values, 'k-');
xlabel('Feature 1');
ylabel('Feature 2');
title('Logistic Regression with Stochastic Gradient Descent (SGD)');
legend('Class 0', 'Class 1', 'Decision Boundary');
hold off;

% Display theta values
disp('Using Stochastic Gradient Descent:');
disp('Coefficients (including intercept):');
disp(theta);

% Prediction function
predict = @(x) round(sigmoid([ones(size(x, 1), 1) x] * theta));

% Compute accuracy
predictions = predict(x(:, 2:end));
accuracy = mean(double(predictions == y)) * 100;
disp(['Training Accuracy: ', num2str(accuracy), '%']);
```

Σχήμα 7: *Logistic Regression with Stochastic Gradient Descent*

Έχουμε τα εξής αποτελέσματα με την εκτέλεση του παραπάνω κώδικα:



Σχήμα 8: *Graph Logistic Regression with Stochastic Gradient Descent*

Το οποίο δίνει ακρίβεια 52.5%. Το ποσοστό αυτό είναι χαμηλό καθώς μας δείχνει ότι το μοντέλο είναι ελαφρώς καλύτερο στο να κατηγοριοποιεί από την κατηγοριοποίηση που θα μας έδινε ένα τίμιο κέρμα. Επομένως δεν θα μέναμε στην παρούσα υλοποίηση εάν θέλαμε να πετύχουμε αυστηρές κατηγοριοποιήσεις.

Task5

Β'. *Logistic Regression* με την μέθοδο *Newton*

Χρησιμοποιόντας την μέθοδο *Newton* με την συνάρτηση:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(-y^{(i)} \log(h_\theta(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right) \tag{3}$$

$$\theta^{(t+1)} = \theta^{(t)} - H^{-1} \nabla_\theta J \tag{4}$$

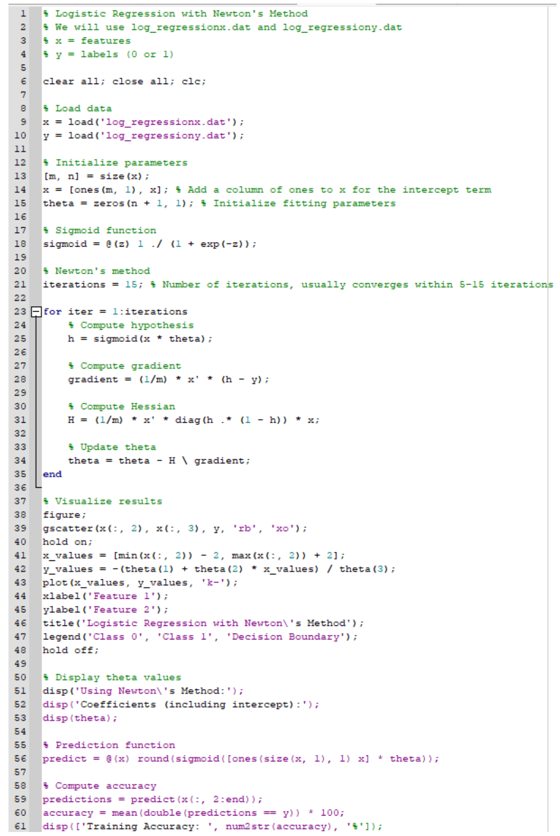
όπου:

$$\nabla_\theta J = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)} \tag{5}$$

και

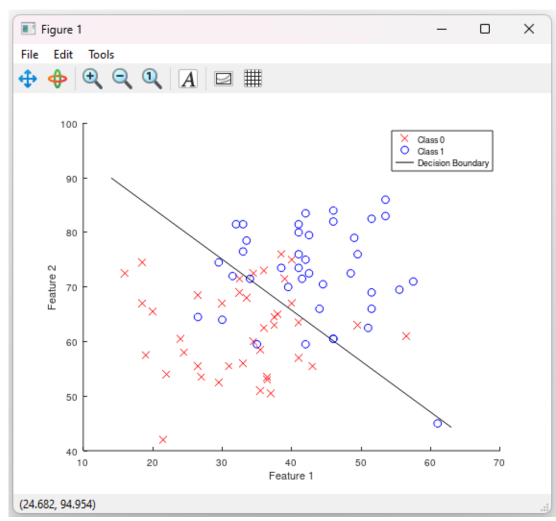
$$H = \frac{1}{m} \sum_{i=1}^{m} \left[h_\theta(x^{(i)})(1 - h_\theta(x^{(i)})) x^{(i)} (x^{(i)})^T \right] \tag{6}$$

Ο κώδικας μας *Task5.m* Μας δίνει τα εξής αποτελέσματα:



```
% Logistic Regression with Newton's Method
% We will use log_regressionx.dat and log_regressiony.dat
% x = features
% y = labels (0 or 1)

clear all; close all; clc;

% Load data
x = load('log_regressionx.dat');
y = load('log_regressiony.dat');

% Initialize parameters
[m, n] = size(x);
x = [ones(m, 1), x]; % Add a column of ones to x for the intercept term
theta = zeros(n + 1, 1); % Initialize fitting parameters

% Sigmoid function
sigmoid = @(z) 1 ./ (1 + exp(-z));

% Newton's method
iterations = 15; % Number of iterations, usually converges within 5-15 iterations

for iter = 1:iterations
    % Compute hypothesis
    h = sigmoid(x * theta);

    % Compute gradient
    gradient = (1/m) * x' * (h - y);

    % Compute Hessian
    H = (1/m) * x' * diag(h .* (1 - h)) * x;

    % Update theta
    theta = theta - H \ gradient;
end

% Visualize results
figure;
gscatter(x(:, 2), x(:, 3), y, 'rb', 'xo');
hold on;
x_values = [min(x(:, 2)) - 2, max(x(:, 2)) + 2];
y_values = -(theta(1) + theta(2) * x_values) / theta(3);
plot(x_values, y_values, 'k-');
xlabel('Feature 1');
ylabel('Feature 2');
title('Logistic Regression with Newton\'s Method');
legend('Class 0', 'Class 1', 'Decision Boundary');
hold off;

% Display theta values
disp('Using Newton\'s Method:');
disp('Coefficients (including intercept):');
disp(theta);

% Prediction function
predict = @(x) round(sigmoid([ones(size(x, 1), 1) x] * theta));

% Compute accuracy
predictions = predict(x(:, 2:end));
accuracy = mean(double(predictions == y)) * 100;
disp(['Training Accuracy: ', num2str(accuracy), '%']);
```

Σχήμα 9: *Logistic Regression with Newton's Method*



Σχήμα 10: *Logistic Regression with Newton's Method Graph*

- Ακρίβεια Εκπαίδευσης: 81.25
- Έλεγχος Σύγκλισης : Σύγκλιση σε 7 επαναλήψεις.

- Τιμές Θήτα
  -16.3787
  0.1483
  0.1589

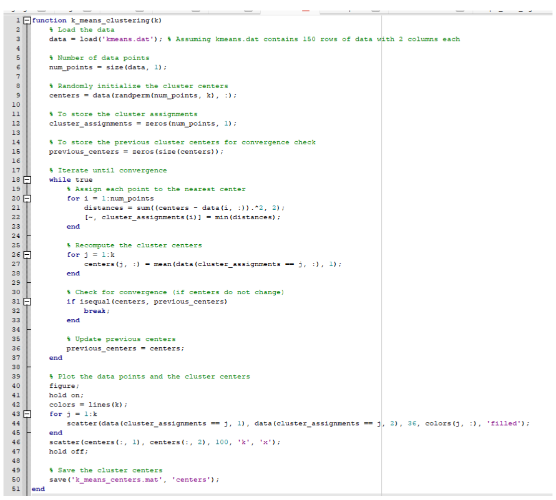
Πράγματι επιτεύχθηκε η σύγκλιση μέσα στο προτεινόμενο όριο των 5 έως 15 επαναλήψεων και μάλιστα πολύ γρήγορα με μόλις 7. Επίσης πολύ καλύτερο ποσοστό ακρίβειας και ακόμα και από την εικόνα βλέπουμε ότι η ευθεία διαχωρίζει πολύ καλύτερα τις δύο κλάσεις, ασφαλώς με ένα ποσοστό σφάλματος.

IV. *Unsupervised Learning*

Task 6

Α′. $K - meansClustering$

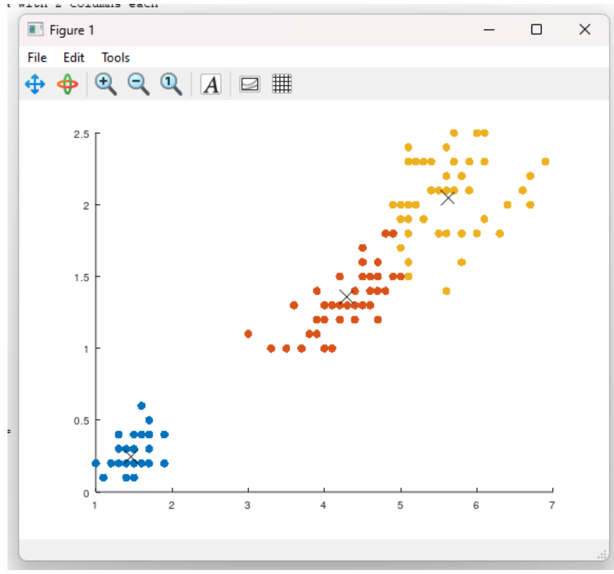
Με την χρήση των αριθμητικών δεδομένων στα οποία θα έχουμε 150 δείγματα και 2 χαρακτηριστικά το καθένα θα δημιουργήσουμε ένα μοντέλο K-*means* για διαφορετικό αριθμό πυρήνων έτσι με την κατασκευή του παρακάτω αλγορίθμου έχουμε:



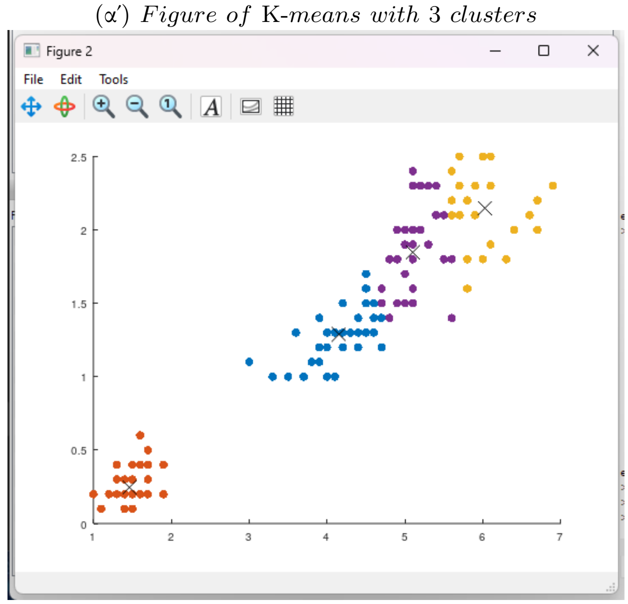
```
function k_means_clustering(k)
    % Load the data
    data = load('kmeans.dat'); % Assuming kmeans.dat contains 150 rows of data with 2 columns each

    % Number of data points
    num_points = size(data, 1);

    % Randomly initialize the cluster centers
    centers = data(randperm(num_points, k), :);

    % To store the cluster assignments
    cluster_assignments = zeros(num_points, 1);

    % To store the previous cluster centers for convergence check
    previous_centers = zeros(size(centers));

    % Iterate until convergence
    while true
        % Assign each point to the nearest center
        for i = 1:num_points
            distances = sum((centers - data(i, :)).^2, 2);
            [~, cluster_assignments(i)] = min(distances);
        end

        % Recompute the cluster centers
        for j = 1:k
            centers(j, :) = mean(data(cluster_assignments == j, :), 1);
        end

        % Check for convergence (if centers do not change)
        if isequal(centers, previous_centers)
            break;
        end

        % Update previous centers
        previous_centers = centers;
    end

    % Plot the data points and the cluster centers
    figure;
    hold on;
    colors = lines(k);
    for j = 1:k
        scatter(data(cluster_assignments == j, 1), data(cluster_assignments == j, 2), 36, colors(j, :), 'filled');
    end
    scatter(centers(:, 1), centers(:, 2), 100, 'k', 'x');
    hold off;

    % Save the cluster centers
    save('k_means_centers.mat', 'centers');
end
```

Σχήμα 11: *Logistic Regression with Newton′s Method Graph*

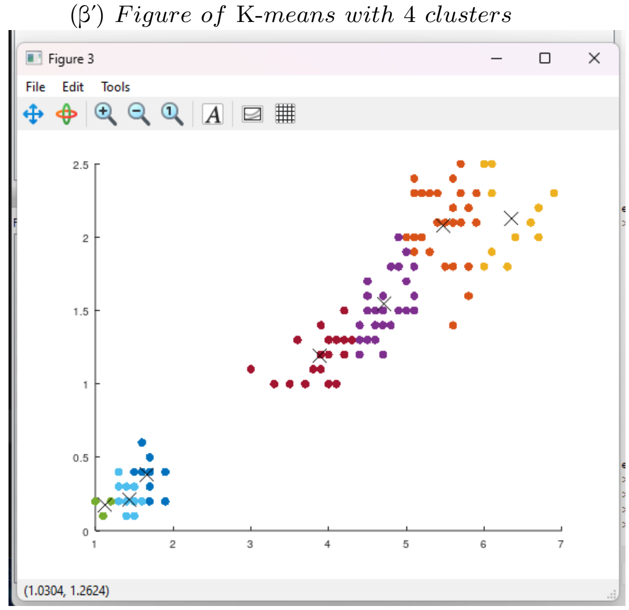
Και αφού τον εκτελέσουμε μπορούμε να εκτελέσουμε $k\_means\_clustering(k)$, όπου $k = 2, 3, 4, \ldots$ και να λάβουμε το αντίστοιχο γράφημα. Δοκιμάζουμε για κ = 3,4 και 7.



(α′) *Figure of* K-*means with* 3 *clusters*



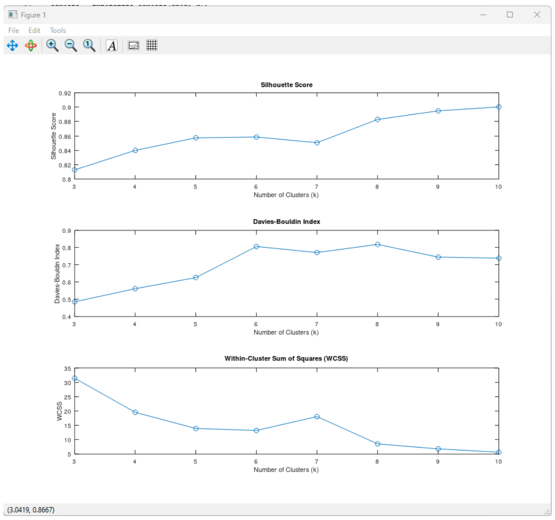
(β′) *Figure of* K-*means with* 4 *clusters*



(γ′) *Figure of* K-*means with* 7 *clusters*

Σχήμα 12: *K_means*

Συγκρίνοντας οπτικά προτιμούμε τους λιγότερους πυρήνες καθώς βλέπουμε ότι στους 7 πυρήνες μια περιοχή που φαίνεται ξεκάθαρα αποκομμένη από τα υπόλοιπα δεδομένα μπορεί εύκολα να αναπαρασταθεί από ένα κέντρο όπως στις πρώτες 2 εικόνες. Για να έχουμε μια καλύτερη εικόνα αναβαθμίζουμε τον κώδικα μας ώστε να βρούμε τον καλύτερο αριθμό πυρήνων. Υλοποιούμε τον $KmeansUltraPlus.m$ και εκτελούμε $evaluate\_k\_means()$. Μπορούμε ευκολά να δώσουμε στον κώδικα να ελέγξει περισσότερους πυρήνες αλλά



Σχήμα 13: $Silhouette's$, $Davies - Bouldin\ Index$, WSS for K $clusters\ where$ κ = [1,10]

Σύμφωνα με τα γραφήματα που φαίνονται στην εικόνα, ο καλύτερος αριθμός των κλάσεων k για τον αλγόριθμο $k - means$ μπορεί να εκτιμηθεί ως εξής:

- $SilhouetteScore$: Ο δείκτης $Silhouette$ αυξάνεται σταθερά μέχρι το $k = 6$ και μετά παραμένει περίπου σταθερός με μικρές διακυμάνσεις. Υψηλότερη τιμή υποδεικνύει καλύτερο $clustering$.
- $Davies - BouldinIndex$: Ο δείκτης $Davies - Bouldin$ είναι χαμηλότερος για $k = 5$, το οποίο υποδεικνύει καλύτερη κατανομή των κλάσεων καθώς χαμηλότερη τιμή υποδεικνύει καλύτερο $clustering$.
- $Within - Cluster\ Sum\ of\ Squares\ (WS)$: Η καμπύλη του WSS παρουσιάζει μεγάλη μείωση μέχρι το $k = 5$ και μετά από εκεί η μείωση είναι μικρότερη. Το σημείο όπου η καμπύλη αρχίζει να γίνεται πιο επίπεδη, γνωστό και ως "$Elbow$", μπορεί να υποδεικνύει τον καλύτερο αριθμό των κλάσεων.

Συμπερασματικά, ο καλύτερος αριθμός των κλάσεων k φαίνεται να είναι είτε $k = 5$ είτε $k = 6$. Ωστόσο, λαμβάνοντας υπόψη τον $Davies - Bouldin\ Index$, το $k = 5$ φαίνεται να είναι η καλύτερη επιλογή.