Regularization and Titanic

Laboratory Exercise 4

Antonio, Daniel

Electronics and Communications Engineering Department

De La Salle University - Manila

Manila, Philippines

Eco, Lance Randall Lim

Electronics and Communications Engineering Department

De La Salle University – Manila

Manila, Philippines

lance\_eco@dlsu.edu.ph

Tan, Sidney Sheldon Urlanda

Electronics and Communications Engineering Department

De La Salle University - Manila

Manila, Philippines

sidney\_tan@dlsu.edu.ph

Zarzoso, Aaron John Pascua

Electronics and Communications Engineering Department

De La Salle University - Manila

Manila, Philippines

aaron\_zarzoso@dlsu.edu.ph

*Abstract*—This document represent the student’s lab report for experiment 4 of the course LBYCP29. It contains the procedures done in the implementation of regularization

Keywords—octave; matlab; regularization; fitting; titanic;

# Introduction

Regularization is a technique used in solving an overfitting problem in statistical models. In addition, it reduces overfitting by adding complexity penalty to the loss function. As the magnitude of the fitting parameters increase, there will be an increase penalty to the cost function. There are two uses of regularization such as linear and logistic regression. This type of technique is used in these regressions since there are so many data that are included which entitles the information to suffer the problem of overfitting which means that the learning is based on memorization instead of a learning of a much more intelligent manner which is what we are trying to achieve. Therefore, we use regularization in order solve this problem especially in processes that consist of a lot of data such as linear regression and logistic regression. Basically, what regularization does is that it adds up additional information in order to minimize overfitting or better yet to prevent if from occurring.

# Instructions/procedure

Part I: Regularized Linear Regression

## Procedure 4.1 Plot the data

* Load the data files "ml4Linx.dat" and "ml4Liny.dat" into your program.
* These correspond to the "x" and "y" variables that you will start out with.
* Notice that in this data, the input "x" is a single feature, so you can plot y as a function of x on a 2-dimensional graph.

## Procedure 4.2

1. Using the Normal equation, find values for θ using the three regularization parameters below:
2. λ = 0 (this is the same case as nonregularized linear regression)
3. λ = 1
4. λ = 10

## Procedure 4.3 Plot the polynomial fit for each value of λ.

* When you have found the answers for θ, verify them with the values in the solutions.
* In addition to listing the values for each element θ\_j of the θ vector, we will also provide the L2-norm of θ so you can quickly check if your answer is correct.
* In Octave, you can calculate the L2-norm of a vector x using the command norm(x).

## Procedure 4.4

* After loading the data, plot the points using different markers to distinguish between the two classifications. The commands in Matlab/Octave will be:

*x = load('ml4Logx.dat');*

*y = load('ml4Logy.dat');*

*figure*

*% Find the indices for the 2 classes*

*pos = find(y);*

*neg = find(y == 0);*

*plot(x(pos, 1), x(pos, 2), '+')*

*hold on*

*plot(x(neg, 1), x(neg, 2), 'o')*

## Procedure 4.5

* Run Newton's Method using the three values of lambda below:

1. λ = 0 (this is the same case as nonregularized logistic regression)
2. λ = 1
3. λ = 10

## Procedure 4.6

* Print out the value of J(θ) during each iteration.
* J(θ) should not be decreasing at any point during Newton's Method.
* If it is, check that you have defined J(θ) correctly.
* Also check your definitions of the gradient and Hessian to make sure there are no mistakes in the regularization parts.

## Procedure 4.7

* After convergence, use your values of theta to find the decision boundary in the classification problem.
* The decision boundary is defined as the line where



## Procedure 4.8

* Finally, because there are 28 elements θ, we will not provide an element-by-element comparison in the solutions.
* Instead, use norm(theta) to calculate the L2- norm of θ, and check it against the norm in the solutions.

Part III Real-world Application

## Application Objective

* To apply regularized logistic regression to predict which passengers survived the Titanic shipwreck tragedy.
* Choose two features from the 'titanic3.xls' data and utilize it in Regularized Logistic Regression Implementation by repeating procedures 4.4 – 4.8.
* Give the necessary plots and analysis for each procedure.

# Data and results

Part I: Regularized Linear Regression

## Procedure 4.1 Plot the data

1. Code:

*- Load the data*

*x = load('ml4Linx.dat');*

*y = load('ml4Liny.dat');*

*- Plot*

*plot(x, y, 'o');*

*xlabel('x');*

*ylabel('y');*

*hold on;*

*[m,n] = size(x);*

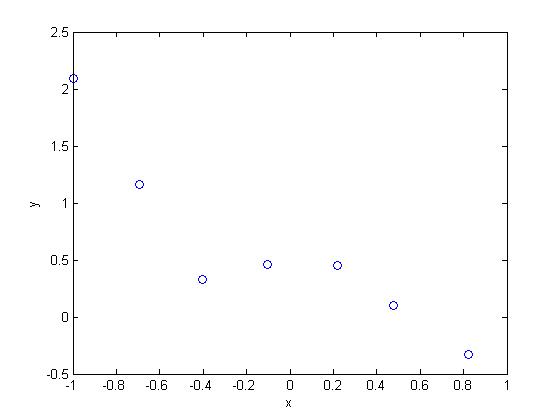


Fig. 1. Plot of x and y data.

## Procedure 4.2

1. Code:

*- 5th order polynomial*

*x = [ones(m,1), x, x.^2, x.^3, x.^4, x.^5];*

*- Initialize theta*

*theta = zeros(size(x(1,:)))';*

*- Set lambda (0, 1, 10)*

*lambda = 0;*

*L = lambda .\* eye(6);*

*L(1) = 0;*

*theta = (x' \* x + L)\(x' \* y);*

Theta values:

a. λ = 0

Theta = [0.472528772874319; 0.681352894856652;

-1.380128418612173; -5.977687467469015; 2.441732684792994; 4.737114334830815]

b. λ = 1

Theta = [0.397595299175466; -0.420666371376896; 0.129592111980193; -0.397473899391432; 0.175255526708740; -0.339387717362337]

c. λ = 10

Theta = [0.520470738359628; -0.182507058295231; 0.060642582038725; -0.148177206219198; 0.074330064766671; -0.127957368751850]

## Procedure 4.3 Plot the polynomial fit for each value of λ.

1. Code:

*- L2 norm*

*theta\_norm = norm(theta);*

*- Set x axis*

*x\_vals = (-1:0.05:1)';*

*- Set regularization line*

*feats = [ones(size(x\_vals)), x\_vals, x\_vals .^2,...x\_vals.^3, x\_vals.^4, x\_vals.^5];*

*- Plot*

*plot(x\_vals, feats \* theta, '--');*

*xlabel('x\_vals');*

*ylabel('features');*

*legend('Training data', 'Lambda = 0');*

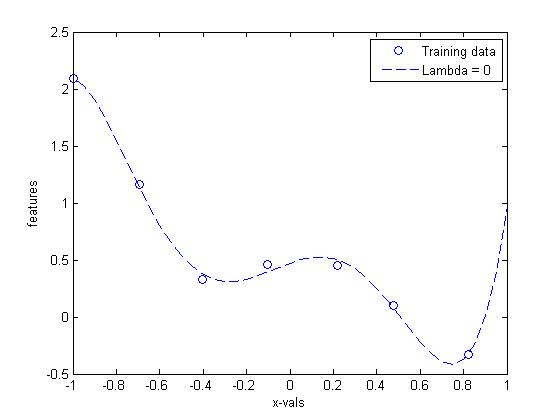


Fig. 2. Plot of the polynomial fit for λ = 0.

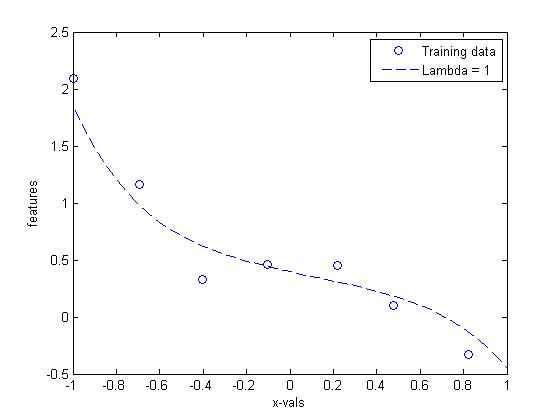


Fig. 3. Plot of the polynomial fit for λ = 1.

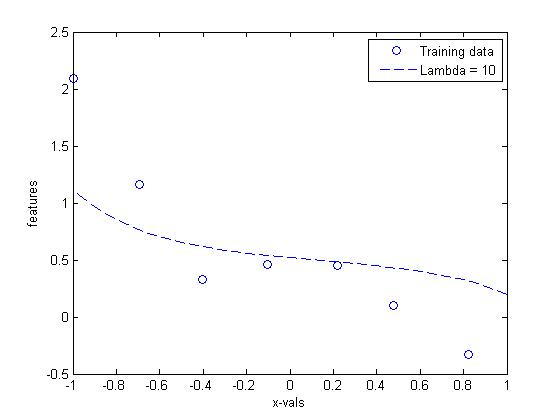


Fig. 4. Plot of the polynomial fit for λ = 10.

Question 1

* From looking at the previous graphs, what conclusions can you make about how the regularization parameter λ affects your model? \_\_\_\_\_It can be concluded that when λ = 0, the model shown is overfitting, λ = 1 shows the right type of model, and λ = 10 shows the underfitting model.\_\_\_\_\_

Part II Regularized logistic regression

## Procedure 4.4

1. Code:

*- Load and plot positive and negative class*

*x = load('ml4Logx.dat');*

*y = load('ml4Logy.dat');*

*pos = find(y == 1);*

*neg = find(y == 0);*

*plot(x(pos,1), x(pos,2), '+');*

*hold on;*

*plot(x(neg,1), x(neg,2), 'o');*

*xlabel('Feature 1');*

*ylabel('Feature 2');*

*legend('y = 1', 'y = 0');*

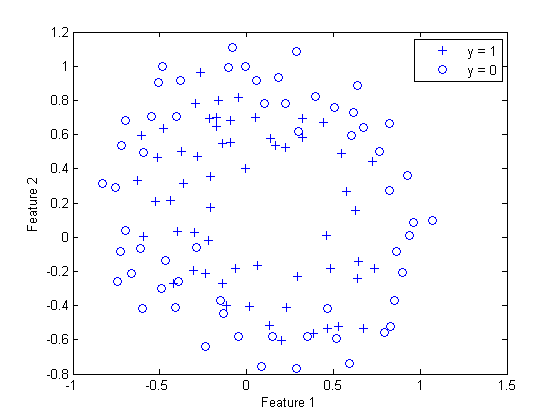


Fig. 5. Plot of the two classifications distinguished by marker type (O and +).

## Procedure 4.5

1. Code:

*- Map feature*

*x = map\_feature(x(:,1),x(:,2));*

*[m,n] = size(x);*

*- Initialize theta*

*theta = zeros(n, 1);*

*- Sigmoid*

*g = inline('1.0 / (1.0 + exp(-z))');*

*- Initialize J and iterations*

*num = 15;*

*J = zeros(num, 1);*

*- Set lambda (0,1,10)*

*lambda = 0;*

*- Newton’s method*

*for i = 1:num*

*z = x \* theta;*

*h = g(z);*

*J(num) =(1/m)\*sum(-y.\*log(h) - (1-y).\*log(1-h)) +...*

*(lambda/(2\*m))\*norm(theta([2:end]))^2;*

*G = (lambda/m).\*theta;*

*G(1) = 0;*

*L = (lambda/m).\*eye(n);*

*L(1) = 0;*

*grad = ((1/m).\*x' \* (h-y)) + G;*

*H = ((1/m).\*x' \* diag(h) \* diag(1-h) \* x) + L;*

*theta = theta - H\grad;*

*end*

## Procedure 4.6

1. Code:

*- Print J for each lambda*

*figure*

*plot(0:MAX\_ITR-1, J, 'o');*

*xlabel('Iteration');*

*ylabel('J');*

a. λ = 0

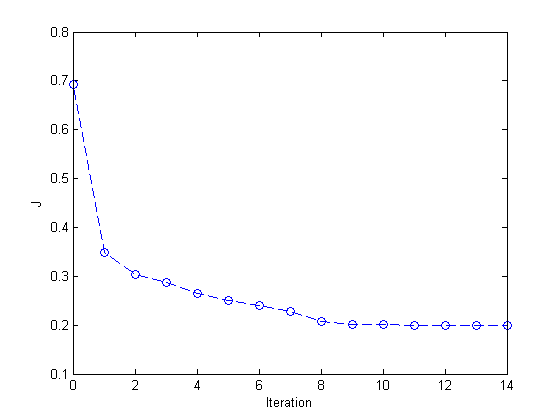


Fig. 6. Plot of iteration and J at λ = 0.

b. λ = 1

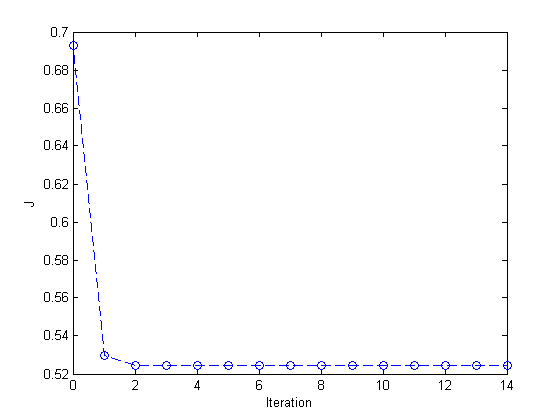


Fig. 7. Plot of iteration and J at λ = 1.

c. λ = 10

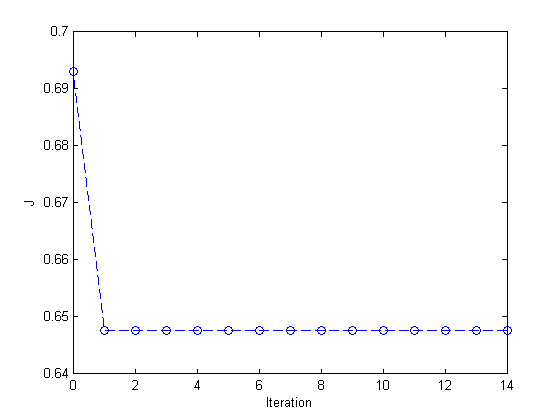


Fig. 8. Plot of iteration and J at λ = 10.

## Procedure 4.7

1. Code:

- Decision boundary line

u = linspace(-1, 1.5, 200);

v = linspace(-1, 1.5, 200);

z = zeros(length(u), length(v));

for i = 1:length(u)

for j = 1:length(v)

z(i,j) = map\_feature(u(i), v(j))\*theta;

end

end

z = z';

contour(u, v, z, [0, 0], 'LineWidth', 2)

legend('y = 1', 'y = 0', 'Decision boundary')

title(sprintf('\\lambda = %g', lambda), 'FontSize', 14)

holf off

a. λ = 0

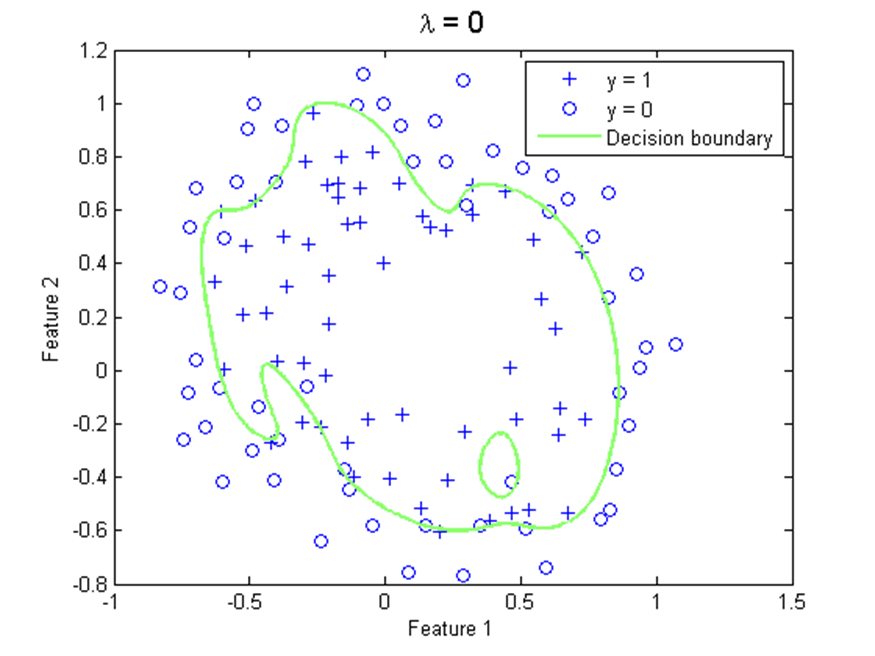


Fig. 9. Plot of iteration and J at λ = 0.

b. λ = 1

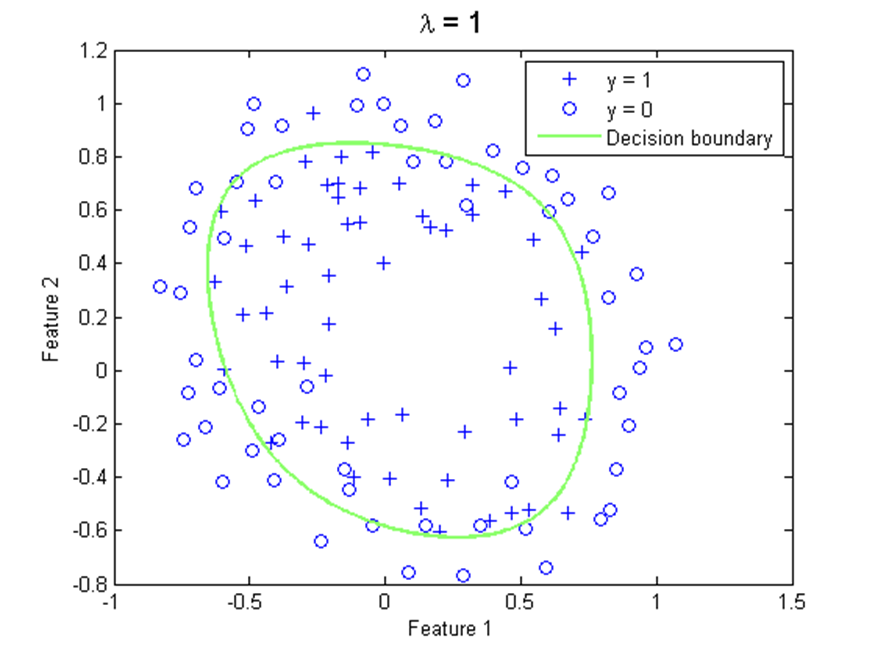


Fig. 10. Plot of iteration and J at λ = 1.

c. λ = 10

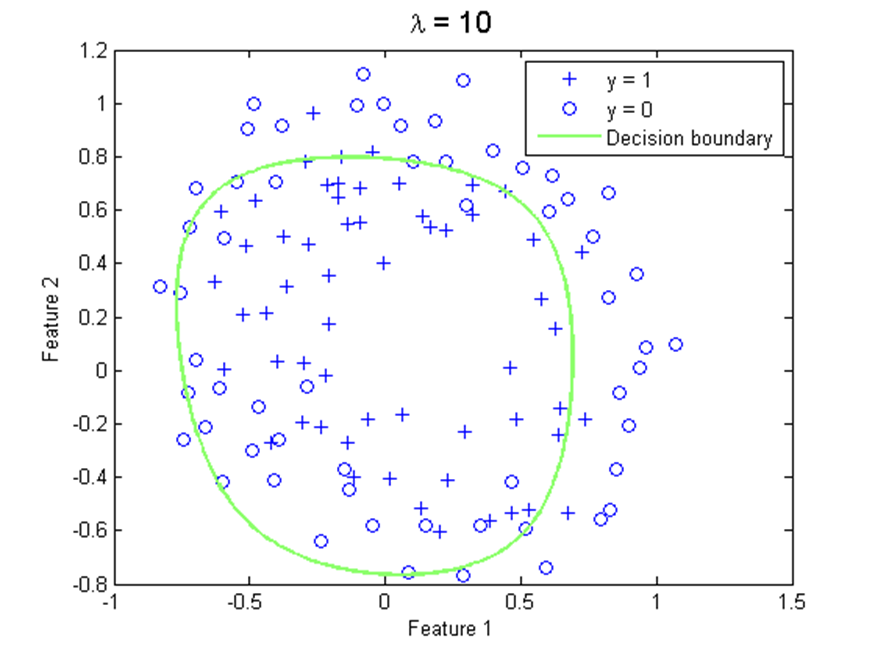


Fig. 11. Plot of iteration and J at λ = 10.

## Procedure 4.8

1. Code:

- L2 norm

theta\_norm = norm(theta);

## Application Objective

Data used from the titanic.xls

x = [pclass; age];

y =[survived];

12077479_1156867787660271_1269853269_n

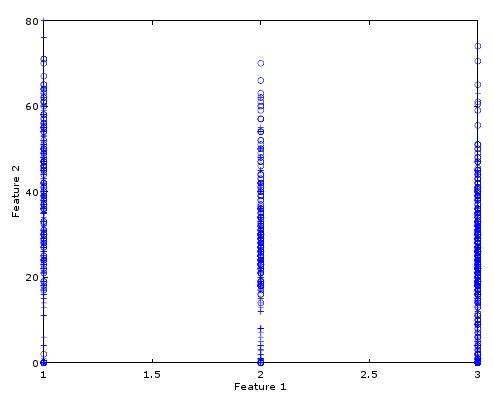


Fig. 12. Plot of the Feature 1 and 2

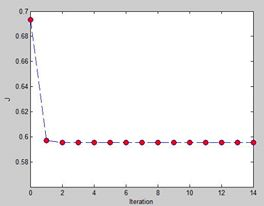


Fig. 13. Block of error from processing titanic data

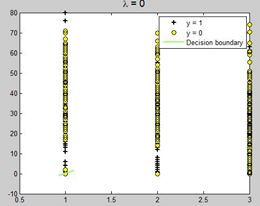


Fig. 14. Plot when λ = 0.

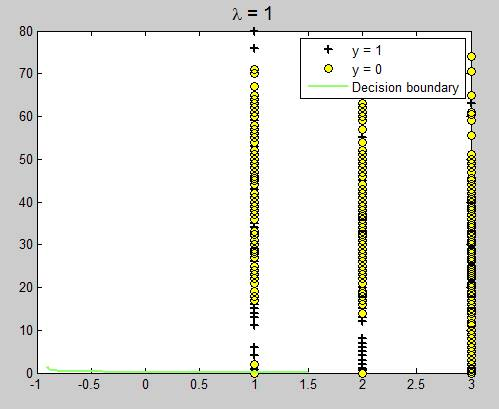


Fig. 15. Plot when λ = 1.

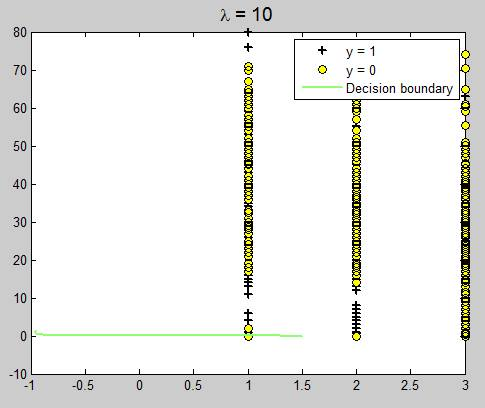


Fig. 16. Plot when λ = 10.

1. Codes:

*% Plot the training data*

*% Use different markers for positives and negatives*

*figure*

*pos = find(y); neg = find(y == 0);*

*plot(x(pos, 1), x(pos, 2), 'k+','LineWidth', 2, 'MarkerSize', 7)*

*hold on*

*plot(x(neg, 1), x(neg, 2), 'ko', 'MarkerFaceColor', 'y', 'MarkerSize', 7)*

*% Add polynomial features to x by*

*% calling the feature mapping function*

*% provided in separate m-file*

*x = map\_feature(x(:,1), x(:,2));*

*[m, n] = size(x);*

*% Initialize fitting parameters*

*theta = zeros(n, 1);*

*% Define the sigmoid function*

*g = inline('1.0 ./ (1.0 + exp(-z))');*

*% setup for Newton's method*

*MAX\_ITR = 15;*

*J = zeros(MAX\_ITR, 1);*

*% Lambda is the regularization parameter*

*lambda = 10;*

*% Newton's Method*

*for i = 1:MAX\_ITR*

*% Calculate the hypothesis function*

*z = x \* theta;*

*h = g(z);*

*% Calculate J (for testing convergence)*

*J(i) =(1/m)\*sum(-y.\*log(h) - (1-y).\*log(1-h))+ ...*

*(lambda/(2\*m))\*norm(theta([2:end]))^2;*

*% Calculate gradient and hessian.*

*G = (lambda/m).\*theta; G(1) = 0; % extra term for gradient*

*L = (lambda/m).\*eye(n); L(1) = 0;% extra term for Hessian*

*grad = ((1/m).\*x' \* (h-y)) + G;*

*H = ((1/m).\*x' \* diag(h) \* diag(1-h) \* x) + L;*

*% Here is the actual update*

*theta = theta - H\grad;*

*end*

*% Show J to determine if algorithm has converged*

*J*

*% display the norm of our parameters*

*norm\_theta = norm(theta)*

*% Plot the results*

*% We will evaluate theta\*x over a*

*% grid of features and plot the contour*

*% where theta\*x equals zero*

*% Here is the grid range*

*u = linspace(-1, 1.5, 200);*

*v = linspace(-1, 1.5, 200);*

*z = zeros(length(u), length(v));*

*% Evaluate z = theta\*x over the grid*

*for i = 1:length(u)*

*for j = 1:length(v)*

*z(i,j) = map\_feature(u(i), v(j))\*theta;*

*end*

*end*

*z = z'; % important to transpose z before calling contour*

*% Plot z = 0*

*% Notice you need to specify the range [0, 0]*

*contour(u, v, z, [0, 0], 'LineWidth', 2)*

*legend('y = 1', 'y = 0', 'Decision boundary')*

*title(sprintf('\\lambda = %g', lambda), 'FontSize', 14)*

*hold off*

*% Uncomment to plot J*

*figure*

*plot(0:MAX\_ITR-1, J, 'o--', 'MarkerFaceColor', 'r', 'MarkerSize', 8)*

*xlabel('Iteration'); ylabel('J')*

# analysis and conclusion

In this experiment, the students was able to make use of the octave software in order to create regularization on both linear and logistic regression. Regularized linear regression is somewhat an approximation but the value of the readings increased compared to the normal linear regression. It focuses on the over fitting or even under fitting which the aim of the regularization is to decrease the value of the cost function with respect to theta. Lambda is the regularization parameter and the penalty of the cost function is directly proportional to the magnitude of the fitting parameter. The students concluded that the value of the lambda controls the data fitting. In addition, the lambda value 1 is the best plot to clearly characterize the positive and the negative.

##### References

1. MathWorks. (N.A.). *MATLAB -* *The Language of Technical Computing*. [Online]. Available: http://www.mathworks.com/  
   products/matlab/
2. Eaton, J. (N.A.)*. About GNU Octave.* [Online]. Available: https://www.gnu.org/software/octave/about.html
3. DLSU. (N.D.). Laboratory Exercise 4: Regularization and Titanic. [File] Available: ML\_Activity4\_Regularization.pdf
4. Andrew Ng. Stanford University, CS 229. Machine LearningCourse Materials. [Online]. Available: http://cs229.stanford.edu/materials

.html