>> data = load('ex1data1.txt');

>> X = data(:, 1); y = data(:, 2);

>> m = length(y);

>> plotData(X, y);

>> X = [ones(m, 1), data(:,1)];

>> theta = zeros(2, 1);

>> iterations = 1500;

>> alpha = 0.01;

>> edit computeCost.m;

>> J = computeCost(X, y, [-1 ; 2]);

>> fprintf('\nWith theta = [-1 ; 2]\nCost computed = %f\n', J);

With theta = [-1 ; 2]

Cost computed = 54.242455

>> fprintf('Expected cost value (approx) 54.24\n');

Expected cost value (approx) 54.24

>> fprintf('Expected cost value (approx) 54.24\n');

Expected cost value (approx) 54.24

>> submit()

== Submitting solutions | Linear Regression with Multiple Variables...

Use token from last successful submission (satyakar3001@gmail.com)? (Y/n): Y

warning: findstr is obsolete; use strfind instead

warning: strmatch is obsolete; use strncmp or strcmp instead

==

== Part Name | Score | Feedback

== --------- | ----- | --------

== Warm-up Exercise | 10 / 10 | Nice work!

== Computing Cost (for One Variable) | 40 / 40 | Nice work!

== Gradient Descent (for One Variable) | 0 / 50 |

== Feature Normalization | 0 / 0 |

== Computing Cost (for Multiple Variables) | 0 / 0 |

== Gradient Descent (for Multiple Variables) | 0 / 0 |

== Normal Equations | 0 / 0 |

== --------------------------------

== | 50 / 100 |

==

>> theta0\_vals = linspace(-10, 10, 100);

>> theta1\_vals = linspace(-1, 4, 100);

>> J\_vals = zeros(length(theta0\_vals), length(theta1\_vals));

>> for i = 1:length(theta0\_vals)

for j = 1:length(theta1\_vals)

t = [theta0\_vals(i); theta1\_vals(j)];

J\_vals(i,j) = computeCost(X, y, t);

end

end

>> J\_vals = J\_vals';

>> figure;

>> surf(theta0\_vals, theta1\_vals, J\_vals)

>> xlabel('\theta\_0'); ylabel('\theta\_1');

>> figure;

>> contour(theta0\_vals, theta1\_vals, J\_vals, logspace(-2, 3, 20))

>> xlabel('\theta\_0'); ylabel('\theta\_1');

>> hold on;

>> plot(theta(1), theta(2), 'rx', 'MarkerSize', 10, 'LineWidth', 2);

>> submit()

== Submitting solutions | Linear Regression with Multiple Variables...

Use token from last successful submission (satyakar3001@gmail.com)? (Y/n): Y

==

== Part Name | Score | Feedback

== --------- | ----- | --------

== Warm-up Exercise | 10 / 10 | Nice work!

== Computing Cost (for One Variable) | 40 / 40 | Nice work!

== Gradient Descent (for One Variable) | 0 / 50 |

== Feature Normalization | 0 / 0 |

== Computing Cost (for Multiple Variables) | 0 / 0 |

== Gradient Descent (for Multiple Variables) | 0 / 0 |

== Normal Equations | 0 / 0 |

== --------------------------------

== | 50 / 100 |

==

>> edit gradientDescent.m;

>> J = computeCost(X,y,theta);

>> J = computeCost(X,y,(-1,2));

parse error:

syntax error

>>> J = computeCost(X,y,(-1,2));

^

>> J = computeCost(X,y,(-1 ; 2));

parse error:

syntax error

>>> J = computeCost(X,y,(-1 ; 2));

^

>> J = computeCost(X,y, [1 ; 2] ;

parse error:

syntax error

>>> J = computeCost(X,y, [1 ; 2] ;

^

>> J = computeCost(X,y, [1 ; 2]) ;

>> theta = gradientDescent(X, y, theta, alpha, iterations);

>> hold on;

>>

>> plot(X(:,2), X\*theta, '-')

>> legend('Training data', 'Linear regression')

>> hold off

>> predict1 = [1, 3.5] \*theta;

>> fprintf('For population = 35,000, we predict a profit of %f\n',...

predict1\*10000);

For population = 35,000, we predict a profit of 4519.767868

>> predict2 = [1, 7] \* theta;

>> fprintf('For population = 70,000, we predict a profit of %f\n',...

predict2\*10000);

For population = 70,000, we predict a profit of 45342.450129

>> fprintf('Program paused. Press enter to continue.\n');

Program paused. Press enter to continue.

>> submit()

== Submitting solutions | Linear Regression with Multiple Variables...

Use token from last successful submission (satyakar3001@gmail.com)? (Y/n): Y

==

== Part Name | Score | Feedback

== --------- | ----- | --------

== Warm-up Exercise | 10 / 10 | Nice work!

== Computing Cost (for One Variable) | 40 / 40 | Nice work!

== Gradient Descent (for One Variable) | 50 / 50 | Nice work!

== Feature Normalization | 0 / 0 |

== Computing Cost (for Multiple Variables) | 0 / 0 |

== Gradient Descent (for Multiple Variables) | 0 / 0 |

== Normal Equations | 0 / 0 |

== --------------------------------

== | 100 / 100 |

==

Ans 2 :- function costFunction();

function J = computeCost(X, y, theta)

%COMPUTECOST Compute cost for linear regression

% J = COMPUTECOST(X, y, theta) computes the cost of using theta as the

% parameter for linear regression to fit the data points in X and y

% Initialize some useful values

m = length(y); % number of training examples

% You need to return the following variables correctly

J = 0;

% ====================== YOUR CODE HERE ======================

% Instructions: Compute the cost of a particular choice of theta

% You should set J to the cost.

i = 1:m;

J = (1/(2\*m)) \* sum( ((theta(1) + theta(2) .\* X(i,2)) - y(i)) .^ 2); % Un-Vectorized

end

ANs 3 :- GradientDescent();

k = 1:m;

t1 = sum((theta(1) + theta(2) .\* X(k,2)) - y(k)); % Un-Vectorized

t2 = sum(((theta(1) + theta(2) .\* X(k,2)) - y(k)) .\* X(k,2)); % Un-Vectorized

theta(1) = theta(1) - (alpha/m) \* (t1);

theta(2) = theta(2) - (alpha/m) \* (t2);

% Save the cost J in every iteration

J\_history(iter) = computeCost(X, y, theta);

% ============================================================

% Save the cost J in every iteration

J\_history(iter) = computeCost(X, y, theta);

end