Programming ex1

1. A simple MATLAB function

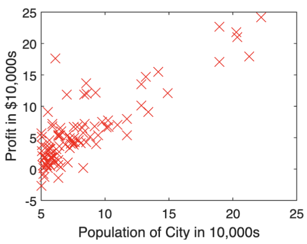
```
A = eye(5);
```

```
ans = 5x5
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
```

2. Linear regression with one variable

2.1 Plotting the data

```
data = load('ex1data1.txt'); % read comma separated data
X = data(:, 1); y = data(:, 2);
plot(x, y, 'rx', 'MarkerSize', 10); % Plot the data
ylabel('Profit in $10,000s'); % Set the y-axis label
xlabel('Population of City in 10,000s'); % Set the x-axis label
```



2.2 Gradient Descent

2.2.1 Update Equations

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

$$h_\theta(x) = \theta^T x = \theta_0 + \theta_1 x_1$$

$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)} \qquad \text{(simultaneously update } \theta_j \text{ for all } j\text{)}$$

2.2.2 Implementation

```
m = length(X); % number of training examples
X = [ones(m,1),data(:,1)]; % Add a column of ones to x
```

```
theta = zeros(2, 1); % initialize fitting parameters
iterations = 1500;
alpha = 0.01;
```

2.2.3 Computing the cost $J(\theta)$

```
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.
J = 1/(2*m) * sum((X * theta - y) .^2);
% =========================================================================
end

% Compute and display initial cost with theta all zeros
computeCost(X, y, theta)
```

```
ans = 32.0727
```

```
% Compute and display initial cost with non-zero theta
computeCost(X, y,[-1; 2])
```

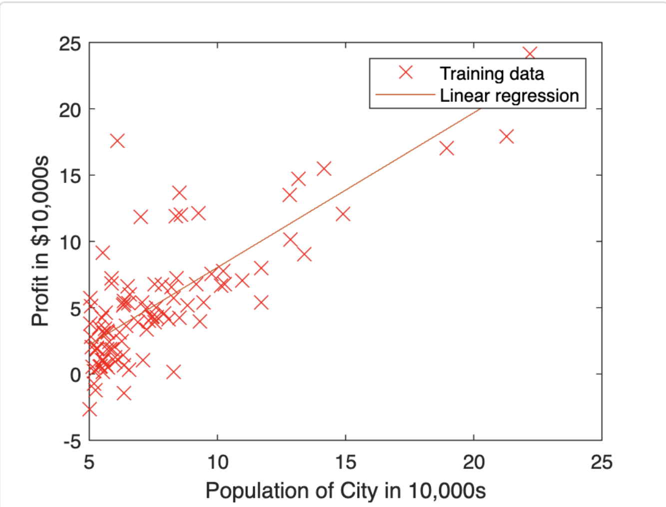
```
ans = 54.2425
```

2.2.4 Gradient descent

```
function [theta, J_history] = gradientDescent(X, y, theta, alpha, num_iters)
%GRADIENTDESCENT Performs gradient descent to learn theta
%   theta = GRADIENTDESCENT(X, y, theta, alpha, num_iters) updates theta by
%   taking num_iters gradient steps with learning rate alpha
% Initialize some useful values
m = length(y); % number of training examples
J_history = zeros(num_iters, 1);
for iter = 1:num_iters
    % ====================== YOUR CODE HERE ======================
    % Instructions: Perform a single gradient step on the parameter vector
    %               theta.
    %
    % Hint: While debugging, it can be useful to print out the values
    %       of the cost function (computeCost) and gradient here.
```

```
    %
    theta = theta - (alpha/m) * (X' * (X * theta - y));
    % ============================================================
    % Save the cost J in every iteration
    J_history(iter) = computeCost(X, y, theta);
end
end

% Run gradient descent:
% Compute theta
theta = gradientDescent(X, y, theta, alpha, iterations);

% Print theta to screen
% Display gradient descent's result
fprintf('Theta computed from gradient descent:\n%f,\n%f',theta(1),theta(2))

% Plot the linear fit
hold on; % keep previous plot visible
plot(X(:,2), X*theta, '-')
legend('Training data', 'Linear regression')
hold off % don't overlay any more plots on this figure

% Predict values for population sizes of 35,000 and 70,000
predict1 = [1, 3.5] *theta;
fprintf('For population = 35,000, we predict a profit of %f\n', predict1*10
predict2 = [1, 7] * theta;
fprintf('For population = 70,000, we predict a profit of %f\n', predict2*10
```

```
Theta computed from gradient descent:
-3.630291,
1.166362
```

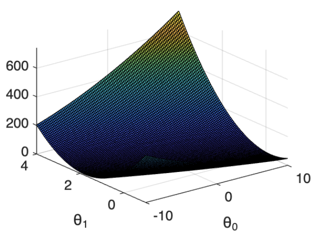


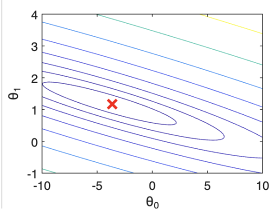

```
For population = 35,000, we predict a profit of 4519.767868
For population = 70,000, we predict a profit of 45342.450129
```

2.4 Visualizing $J(\theta)$

```
% Visualizing J(theta_0, theta_1):
% Grid over which we will calculate J
theta0_vals = linspace(-10, 10, 100);
theta1_vals = linspace(-1, 4, 100);
```

```

% initialize J_vals to a matrix of 0's
J_vals = zeros(length(theta0_vals), length(theta1_vals));

% Fill out J_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

% Because of the way meshgrids work in the surf command, we need to
% transpose J_vals before calling surf, or else the axes will be flipped
J_vals = J_vals';

% Surface plot
figure;
surf(theta0_vals, theta1_vals, J_vals)
xlabel('\theta_0'); ylabel('\theta_1');

% Contour plot
figure;
% Plot J_vals as 15 contours spaced logarithmically between 0.01 and 100
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);
hold off;

```





3. Linear regression with multiple variables

```
% Load Data
data = load('ex1data2.txt');
X = data(:, 1:2);
y = data(:, 3);
m = length(y);

% Print out some data points
% First 10 examples from the dataset
```

```
fprintf(' x = [%.0f %.0f], y = %.0f \n', [X(1:10,:) y(1:10,:)]');
```

```
x = [2104 3], y = 399900
x = [1600 3], y = 329900
x = [2400 3], y = 369000
x = [1416 2], y = 232000
x = [3000 4], y = 539900
x = [1985 4], y = 299900
x = [1534 3], y = 314900
x = [1427 3], y = 198999
x = [1380 3], y = 212000
x = [1494 3], y = 242500
```

3.1 Feature Normalization

```
function [X_norm, mu, sigma] = featureNormalize(X)
%FEATURENORMALIZE Normalizes the features in X
%   FEATURENORMALIZE(X) returns a normalized version of X where
%   the mean value of each feature is 0 and the standard deviation
%   is 1. This is often a good preprocessing step to do when
%   working with learning algorithms.
% You need to set these values correctly
X_norm = X;
mu = zeros(1, size(X, 2));
sigma = zeros(1, size(X, 2));
% ====================== YOUR CODE HERE ======================
% Instructions: First, for each feature dimension, compute the mean
%               of the feature and subtract it from the dataset,
%               storing the mean value in mu. Next, compute the
%               standard deviation of each feature and divide
%               each feature by it's standard deviation, storing
%               the standard deviation in sigma.
%
%               Note that X is a matrix where each column is a
%               feature and each row is an example. You need
%               to perform the normalization separately for
%               each feature.
%
% Hint: You might find the 'mean' and 'std' functions useful.
%
mu = mean(X_norm);
sigma = std(X_norm);
for i = 1 : size(X, 2)
    X_norm(:, i) = X_norm(:, i) - mu(1, i);
    X_norm(:, i) = X_norm(:, i) ./ sigma(1, i);
end
% ============================================================
end
```

```
% Scale features and set them to zero mean
[X, mu, sigma] = featureNormalize(X);
```

```
% Add intercept term to X
X = [ones(m, 1) X];
```

3.2 Gradient Descent

$$J(\theta) = \frac{1}{2m}\left(X\theta - \vec{y}\right)^T\left(X\theta - \vec{y}\right)$$

$$X = \begin{bmatrix} - \left(x^{(1)}\right)^T - \\ - \left(x^{(2)}\right)^T - \\ \vdots \\ - \left(x^{(m)}\right)^T - \end{bmatrix} \qquad \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

```
function [theta, J_history] = gradientDescentMulti(X, y, theta, alpha, num_
%GRADIENTDESCENTMULTI Performs gradient descent to learn theta
%   theta = GRADIENTDESCENTMULTI(x, y, theta, alpha, num_iters) updates thet
%   taking num_iters gradient steps with learning rate alpha
% Initialize some useful values
m = length(y); % number of training examples
J_history = zeros(num_iters, 1);
for iter = 1:num_iters
    % ====================== YOUR CODE HERE ======================
    % Instructions: Perform a single gradient step on the parameter vector
    %               theta.
    %
    % Hint: While debugging, it can be useful to print out the values
    %       of the cost function (computeCostMulti) and gradient here.
    %
 theta = theta - (alpha/m) * (X' * (X * theta - y));
    % ============================================================
    % Save the cost J in every iteration
    J_history(iter) = computeCostMulti(X, y, theta);
end
end
```

```
function J = computeCostMulti(X, y, theta)
%COMPUTECOSTMULTI Compute cost for linear regression with multiple variables
%   J = COMPUTECOSTMULTI(X, y, theta) computes the cost of using theta as th
%   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.
J = 1/(2*m) * sum((X * theta - y) .^2);
% =========================================================================
```

```
end
```

```
% Run gradient descent
% Choose some alpha value
alpha = 0.1;
num_iters = 400;

% Init Theta and Run Gradient Descent
theta = zeros(3, 1);
[theta, ~] = gradientDescentMulti(X, y, theta, alpha, num_iters);

% Display gradient descent's result
fprintf('Theta computed from gradient descent:\n%f\n%f\n%f',theta(1),theta(2
```

```
Theta computed from gradient descent:
340412.659574
110631.048958
-6649.472950
```

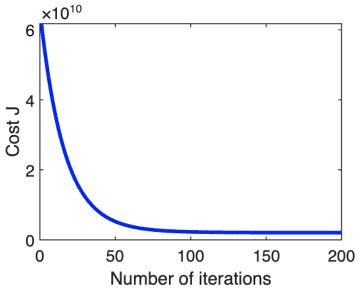
```
% Estimate the price of a 1650 sq-ft, 3 br house
% ====================== YOUR CODE HERE ======================

param = [1650, 3];
norm_param = (param - mu) ./sigma;
real_param = [1, norm_param];



price = real_param * theta; % Enter your price formula here

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

fprintf('Predicted price of a 1650 sq-ft, 3 br house (using gradient descent
```

```
Predicted price of a 1650 sq-ft, 3 br house (using gradient descent):
 $293081.464622
```

3.2.1 Optional (ungraded) exercise: Selecting learning rates

```
% Run gradient descent:
% Choose some alpha value
alpha = 0.03;
num_iters = 200;

% Init Theta and Run Gradient Descent
theta = zeros(3, 1);
[~, J_history] = gradientDescentMulti(X, y, theta, alpha, num_iters);

% Plot the convergence graph
plot(1:num_iters, J_history, '-b', 'LineWidth', 2);
xlabel('Number of iterations');
```

```
ylabel('Cost J');
```




```
% Run gradient descent
% Replace the value of alpha below best alpha value you found above
alpha = 0.03;
num_iters = 200;

% Init Theta and Run Gradient Descent
theta = zeros(3, 1);
[theta, ~] = gradientDescentMulti(X, y, theta, alpha, num_iters);

% Display gradient descent's result
fprintf('Theta computed from gradient descent:\n%f\n%f\n%f',theta(1),theta(2

% Estimate the price of a 1650 sq-ft, 3 br house. You can use the same
% code you entered ealier to predict the price
% ===================== YOUR CODE HERE ======================
param = [1650, 3];
norm_param = (param - mu) ./sigma;
real_param = [1, norm_param];


price = real_param * theta;
% ============================================================

fprintf('Predicted price of a 1650 sq-ft, 3 br house (using gradient descent
```

```
Theta computed from gradient descent:
339642.904508
106274.973805
-2302.198941
```

```
Predicted price of a 1650 sq-ft, 3 br house (using gradient descent):
 $293261.551453
```

3.3 Normal Equations

$$\theta = \left(X^T X\right)^{-1} X^T \vec{y}$$

```
function [theta] = normalEqn(X, y)
%NORMALEQN Computes the closed-form solution to linear regression
%   NORMALEQN(X,y) computes the closed-form solution to linear
%   regression using the normal equations.
theta = zeros(size(X, 2), 1);
```

```
% ===================== YOUR CODE HERE ======================
% Instructions: Complete the code to compute the closed form solution
%               to linear regression and put the result in theta.
%
% ---------------------- Sample Solution ----------------------
theta = pinv(X' * X) * X' * y;
% -------------------------------------------------------------
% ============================================================
end
```

```
% Solve with normal equations:
% Load Data
data = csvread('ex1data2.txt');
X = data(:, 1:2);
y = data(:, 3);
m = length(y);

% Add intercept term to X
X = [ones(m, 1) X];

% Calculate the parameters from the normal equation
theta = normalEqn(X, y);

% Display normal equation's result
fprintf('Theta computed from the normal equations:\n%f\n%f\n%f', theta(1),th
```

```
Theta computed from the normal equations:
89597.909544
139.210674
-8738.019113
```

```
% Estimate the price of a 1650 sq-ft, 3 br house.
% ===================== YOUR CODE HERE ======================
param = [1650, 3];
real_param = [1, param];
price = real_param * theta; % Enter your price forumla here

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

fprintf('Predicted price of a 1650 sq-ft, 3 br house (using normal equations
```

```
Predicted price of a 1650 sq-ft, 3 br house (using normal equations):
 $293081.464335
```