**Machine Learning week 1 Programming Excercise**

https://blog.csdn.net/yunlong34574/article/details/8869108

1 Octave 中的for 和 while

for i<10,

i = i + 1;

end;

这样是错误的，这里的for 应该换成while。

2 Octave 中的std和mean

函数std(x)，算出x的标准偏差。x可以是一行的matrix或者一个多行matrix，如果只有一行，那么就是算一行的标准偏差，如果有多行，就是算每一列的标准偏差。  
std（x，a）也是x的标准偏差，但是a可以＝0或者1.如果是0和前面没有区别，如果是1就是最后除以n，而不是n－1.（你参考计算标准偏差的公式，一般都用除以n－1的公式。）  
std (x, a, b) 这里a表示是要用n还是n－1，如果是a是0就是除以n－1，如果是1就是除以n，b这里是维数，比如说  
1 2 3 4；  
4 5 6 1；  
如果b是1，就是按照列分，如果b是2就是按照行分，如果是三维的矩阵，b＝3就按照第三维来分数据

M = mean(A)，如果A只有一行或者一列数据，那么计算结果为这一行或这一列的平均值，如果A是一个矩阵，那么这里默认按照列来计算平均值  
  M = mean(A,dim)，dim=1 表示按照列计算，dim=2表示按照行计算

Linear regression with one variable

ex1data1.txt 下面是部分数据，第一列是城市人口数量，第二列是在此城市开分店的利润

1. 6.1101,17.592
2. 5.5277,9.1302
3. 8.5186,13.662
4. 7.0032,11.854
5. 5.8598,6.8233
6. 8.3829,11.886
7. 7.4764,4.3483

主程序

1. %% Machine Learning Online Class - Exercise 1: Linear Regression
3. % Instructions
4. % ------------
5. %
6. % This file contains code that helps you get started on the
7. % linear exercise. You will need to complete the following functions
8. % in this exericse:
9. %
10. % warmUpExercise.m
11. % plotData.m
12. % gradientDescent.m
13. % computeCost.m
14. % gradientDescentMulti.m
15. % computeCostMulti.m
16. % featureNormalize.m
17. % normalEqn.m
18. %
19. % For this exercise, you will not need to change any code in this file,
20. % or any other files other than those mentioned above.
21. %
22. % x refers to the population size in 10,000s
23. % y refers to the profit in $10,000s
24. %
26. %% Initialization
27. clear ; close all; clc
29. %% ==================== Part 1: Basic Function ====================
30. % Complete warmUpExercise.m
31. fprintf('Running warmUpExercise ... \n');
32. fprintf('5x5 Identity Matrix: \n');
33. warmUpExercise()
35. fprintf('Program paused. Press enter to continue.\n');
36. pause;

39. %% ======================= Part 2: Plotting =======================
40. fprintf('Plotting Data ...\n')
41. data = load('ex1data1.txt');
42. X = data(:, 1); y = data(:, 2);
43. m = length(y); % number of training examples
45. % Plot Data
46. % Note: You have to complete the code in plotData.m
47. plotData(X, y);
49. fprintf('Program paused. Press enter to continue.\n');
50. pause;
52. %% =================== Part 3: Gradient descent ===================
53. fprintf('Running Gradient Descent ...\n')
55. X = [ones(m, 1), data(:,1)]; % Add a column of ones to x
56. theta = zeros(2, 1); % initialize fitting parameters
58. % Some gradient descent settings
59. iterations = 1500;
60. alpha = 0.01;
62. % compute and display initial cost
63. computeCost(X, y, theta)
64. % run gradient descent
65. theta = gradientDescent(X, y, theta, alpha, iterations);
67. % print theta to screen
68. fprintf('Theta found by gradient descent: ');
69. fprintf('%f %f \n', theta(1), theta(2));
71. % Plot the linear fit
72. hold on; % keep previous plot visible
73. plot(X(:,2), X\*theta, '-')
74. legend('Training data', 'Linear regression')
75. hold off % don't overlay any more plots on this figure
76. % Predict values for population sizes of 35,000 and 70,000
77. predict1 = [1, 3.5] \*theta;
78. fprintf('For population = 35,000, we predict a profit of %f\n',...
79. predict1\*10000);
80. predict2 = [1, 7] \* theta;
81. fprintf('For population = 70,000, we predict a profit of %f\n',...
82. predict2\*10000);
84. fprintf('Program paused. Press enter to continue.\n');
85. pause;
87. %% ============= Part 4: Visualizing J(theta\_0, theta\_1) =============
88. fprintf('Visualizing J(theta\_0, theta\_1) ...\n')
90. % Grid over which we will calculate J
91. theta0\_vals = linspace(-10, 10, 100);
92. theta1\_vals = linspace(-1, 4, 100);
94. % initialize J\_vals to a matrix of 0's
95. J\_vals = zeros(length(theta0\_vals), length(theta1\_vals));
97. % Fill out J\_vals
98. for i = 1:length(theta0\_vals)
99. for j = 1:length(theta1\_vals)
100. t = [theta0\_vals(i); theta1\_vals(j)];
101. J\_vals(i,j) = computeCost(X, y, t);
102. end
103. end

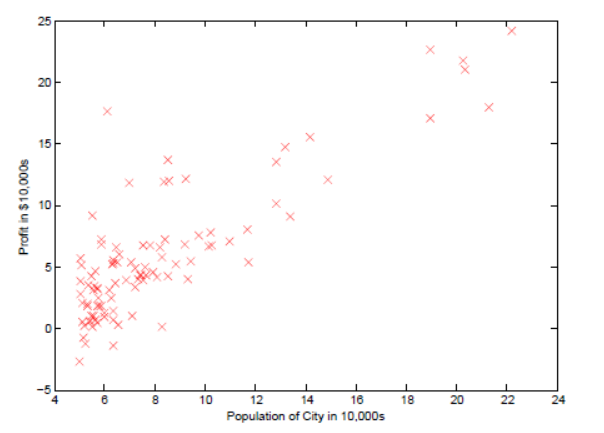
106. % Because of the way meshgrids work in the surf command, we need to
107. % transpose J\_vals before calling surf, or else the axes will be flipped
108. J\_vals = J\_vals';
109. % Surface plot
110. figure;
111. surf(theta0\_vals, theta1\_vals, J\_vals)
112. xlabel('\theta\_0'); ylabel('\theta\_1');
114. % Contour plot
115. figure;
116. % Plot J\_vals as 15 contours spaced logarithmically between 0.01 and 100
117. contour(theta0\_vals, theta1\_vals, J\_vals, logspace(-2, 3, 20))
118. xlabel('\theta\_0'); ylabel('\theta\_1');
119. hold on;
120. plot(theta(1), theta(2), 'rx', 'MarkerSize', 10, 'LineWidth', 2);

1 ploting data

1. function plotData(x, y)
2. %PLOTDATA Plots the data points x and y into a new figure
3. % PLOTDATA(x,y) plots the data points and gives the figure axes labels of
4. % population and profit.
5. % ====================== YOUR CODE HERE ======================
6. % Instructions: Plot the training data into a figure using the
7. % "figure" and "plot" commands. Set the axes labels using
8. % the "xlabel" and "ylabel" commands. Assume the
9. % population and revenue data have been passed in
10. % as the x and y arguments of this function.
11. %
12. % Hint: You can use the 'rx' option with plot to have the markers
13. % appear as red crosses. Furthermore, you can make the
14. % markers larger by using plot(..., 'rx', 'MarkerSize', 10);
16. figure; % open a new figure window
18. plot(x, y, 'rx', 'MarkerSize', 10);
19. ylabel('Profit in $10,000s');
20. xlabel('Population of City in 10,000s');



25. % ============================================================
27. end



2 Gradient Descent

cost function

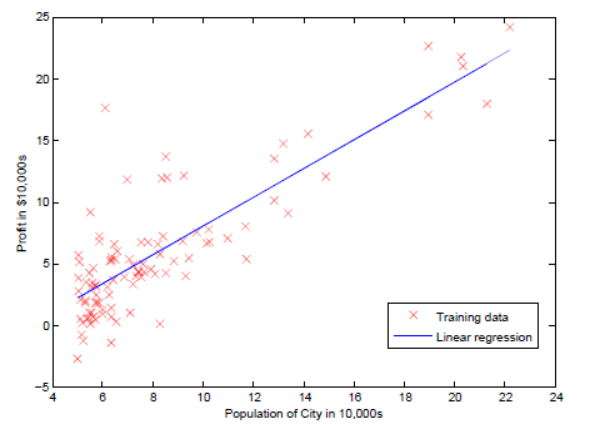
1. function J = computeCost(X, y, theta)
2. %COMPUTECOST Compute cost for linear regression
3. % J = COMPUTECOST(X, y, theta) computes the cost of using theta as the
4. % parameter for linear regression to fit the data points in X and y
6. % Initialize some useful values
7. m = length(y); % number of training examples
9. % You need to return the following variables correctly
10. J = 0;
12. % ====================== YOUR CODE HERE ======================
13. % Instructions: Compute the cost of a particular choice of theta
14. % You should set J to the cost.
15. J = sum((X \* theta - y).^2)/(2\*m);

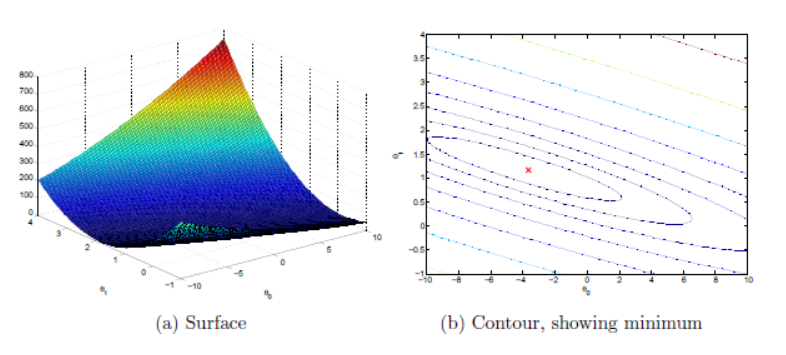


20. % =========================================================================
22. end

Gradient Descent

1. function [theta, J\_history] = gradientDescent(X, y, theta, alpha, num\_iters)
2. %GRADIENTDESCENT Performs gradient descent to learn theta
3. % theta = GRADIENTDESENT(X, y, theta, alpha, num\_iters) updates theta by
4. % taking num\_iters gradient steps with learning rate alpha
6. % Initialize some useful values
7. m = length(y); % number of training examples
8. J\_history = zeros(num\_iters, 1);
10. for iter = 1:num\_iters
12. % ====================== YOUR CODE HERE ======================
13. % Instructions: Perform a single gradient step on the parameter vector
14. % theta.
15. %
16. % Hint: While debugging, it can be useful to print out the values
17. % of the cost function (computeCost) and gradient here.
18. %
19. % ============================================================
21. % Save the cost J in every iteration
22. theta = theta - alpha \* (X' \* (X \* theta - y)) / m;
23. J\_history(iter) = computeCost(X, y, theta);
25. end
27. end

3 Visualization在主程序的part 4中通过surf方法和contour方法分别画出来cost function随着theta1和theta2的变化情况以及等高线  




总结：

1 一开始计算cost function的时候忘记1/2m的系数了，导致计算结果是一个很大的值

2 最后一步中画surf的方法可以借鉴

3 一个很好的习惯就是计算grandient descent的时候，跟踪J(theta)的变化，来分析cost function随着迭代的收敛情况，来决定选择合适的alpha学习速率。

4 学习X = [ones(m, 1), data(:,1)]; % Add a column of ones to x， 其中为feature增加一列1的方法

Linear regression with multiple variables

1 dataset 第一列是房屋的面积，第二列是房屋中包含的卧室的数量，第三列是目标结果，在这里是价格

1. 2104,3,399900
2. 1600,3,329900
3. 2400,3,369000
4. 1416,2,232000
5. 3000,4,539900
6. 1985,4,299900
7. 1534,3,314900
8. 1427,3,198999

1 Feature Normalization  
很容易发现，第一列和第二列数据的量级相差很大，不利于梯度下降的收敛，这里对数据进行规约，采取的方法是

（sample - mean）/ standard divation, 另外一种方案是（sample - mean）/ （max - min），这里采取第一种

使用的函数分别为mean和std计算平均值和标准差

1. function [X\_norm, mu, sigma] = featureNormalize(X)
2. %FEATURENORMALIZE Normalizes the features in X
3. % FEATURENORMALIZE(X) returns a normalized version of X where
4. % the mean value of each feature is 0 and the standard deviation
5. % is 1. This is often a good preprocessing step to do when
6. % working with learning algorithms.
8. % You need to set these values correctly
9. X\_norm = X;
10. mu = zeros(1, size(X, 2));
11. sigma = zeros(1, size(X, 2));
13. % ====================== YOUR CODE HERE ======================
14. % Instructions: First, for each feature dimension, compute the mean
15. % of the feature and subtract it from the dataset,
16. % storing the mean value in mu. Next, compute the
17. % standard deviation of each feature and divide
18. % each feature by it's standard deviation, storing
19. % the standard deviation in sigma.
20. %
21. % Note that X is a matrix where each column is a
22. % feature and each row is an example. You need
23. % to perform the normalization separately for
24. % each feature.
25. %
26. % Hint: You might find the 'mean' and 'std' functions useful.
27. %
28. mu = mean(X,1);
29. sigma = std(X);
30. i = 1;
31. le = size(X, 2);
32. while i <= le,
33. X\_norm(:,i) = (X(:,i) - mu(1,i))/sigma(1,i);
34. i = i + 1;
35. end;

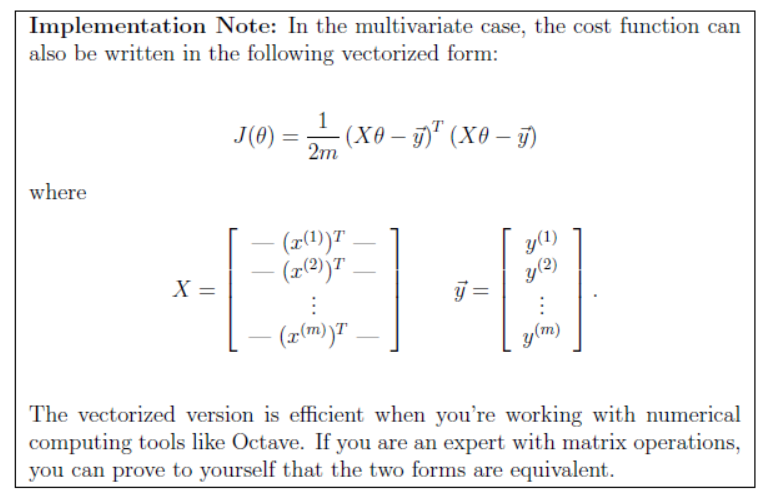



41. % ============================================================
43. end

这里计算得到的mean和std应该保存下来，当需要对新数据进行预测的时候，使用这里计算得到的mean和std来进行数据规约，然后根据theta的值进行预测和估算

2 Gradient Descent

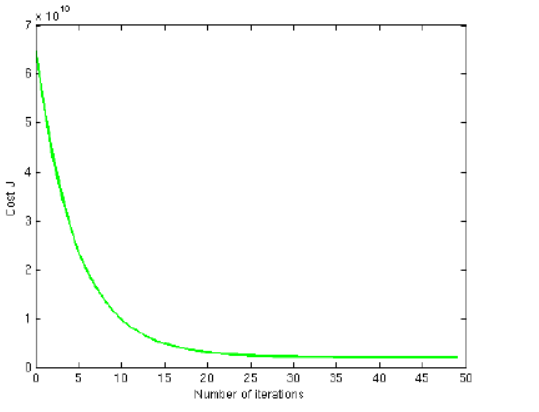
这里和上面的代码保持一致，但是有另外的方法计算，稍微一点不同作为参考



3 Selecting Learning Rate

We recommend trying values of the learning rate on a log-scale, at multiplicative steps of about 3 times the previous value (i.e., 0.3, 0.1, 0.03, 0.01 and so on).

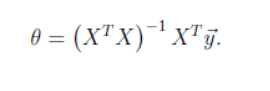
把J(theta)在每次迭代的变化情况画出来后可以得到下面的图片，由此可以看出来学习速率alpha是可以工作的，可以尝试把alpha调大，本来alpha默认是0.03后来我将它调成1之后，J(theta)快速的收敛到了最小值。



总结： 当把数据做了标准化之后，得到theta，在进行预测新的数据的时候，要再次把新数据进行规约，然后计算

Normal Equation

这里非常简单的套用公式即可



1. function [theta] = normalEqn(X, y)
2. %NORMALEQN Computes the closed-form solution to linear regression
3. % NORMALEQN(X,y) computes the closed-form solution to linear
4. % regression using the normal equations.
6. theta = zeros(size(X, 2), 1);
8. % ====================== YOUR CODE HERE ======================
9. % Instructions: Complete the code to compute the closed form solution
10. % to linear regression and put the result in theta.
11. %
13. % ---------------------- Sample Solution ----------------------
14. theta = pinv(X'\*X)\*X'\*y;
16. % -------------------------------------------------------------

19. % ============================================================
21. end

在这里normal equation可以一次计算就得到最优结果，经过比较发现normal equation和gradient descent计算的结果不一样，其实原因是因为上面的gradient descent做过的数据规约的处理，所以会得到不一样的theta，但是两种方法预测的新数据时是可以得到相同的结果。

附上multivariable情况下的主程序

1. %% Machine Learning Online Class
2. % Exercise 1: Linear regression with multiple variables
3. %
4. % Instructions
5. % ------------
6. %
7. % This file contains code that helps you get started on the
8. % linear regression exercise.
9. %
10. % You will need to complete the following functions in this
11. % exericse:
12. %
13. % warmUpExercise.m
14. % plotData.m
15. % gradientDescent.m
16. % computeCost.m
17. % gradientDescentMulti.m
18. % computeCostMulti.m
19. % featureNormalize.m
20. % normalEqn.m
21. %
22. % For this part of the exercise, you will need to change some
23. % parts of the code below for various experiments (e.g., changing
24. % learning rates).
25. %
26. %% Initialization
27. %% ================ Part 1: Feature Normalization ================
29. %% Clear and Close Figures
30. clear ; close all; clc
32. fprintf('Loading data ...\n');
34. %% Load Data
35. data = load('ex1data2.txt');
36. X = data(:, 1:2);
37. y = data(:, 3);
38. m = length(y);
40. % Print out some data points
41. fprintf('First 10 examples from the dataset: \n');
42. fprintf(' x = [%.0f %.0f], y = %.0f \n', [X(1:10,:) y(1:10,:)]');
43. fprintf('Program paused. Press enter to continue.\n');
44. pause;
46. % Scale features and set them to zero mean
47. fprintf('Normalizing Features ...\n');
49. [X mu sigma] = featureNormalize(X);
51. % Add intercept term to X
52. X = [ones(m, 1) X];

55. %% ================ Part 2: Gradient Descent ================
57. % ====================== YOUR CODE HERE ======================
58. % Instructions: We have provided you with the following starter
59. % code that runs gradient descent with a particular
60. % learning rate (alpha).
61. %
62. % Your task is to first make sure that your functions -
63. % computeCost and gradientDescent already work with
64. % this starter code and support multiple variables.
65. %
66. % After that, try running gradient descent with
67. % different values of alpha and see which one gives
68. % you the best result.
69. %
70. % Finally, you should complete the code at the end
71. % to predict the price of a 1650 sq-ft, 3 br house.
72. %
73. % Hint: By using the 'hold on' command, you can plot multiple
74. % graphs on the same figure.
75. %
76. % Hint: At prediction, make sure you do the same feature normalization.
77. %
79. fprintf('Running gradient descent ...\n');
81. % Choose some alpha value
82. alpha = 0.01;
83. num\_iters = 400;
85. % Init Theta and Run Gradient Descent
86. theta = zeros(3, 1);
87. [theta, J\_history] = gradientDescentMulti(X, y, theta, 1, num\_iters);
89. % Plot the convergence graph
90. figure;
91. plot(1:50, J\_history(1:50), '-b', 'LineWidth', 2);
92. xlabel('Number of iterations');
93. ylabel('Cost J');
95. % Display gradient descent's result
96. fprintf('Theta computed from gradient descent: \n');
97. fprintf(' %f \n', theta);
98. fprintf('\n');
100. % Estimate the price of a 1650 sq-ft, 3 br house
101. % ====================== YOUR CODE HERE ======================
102. % Recall that the first column of X is all-ones. Thus, it does
103. % not need to be normalized.
104. price = ([1 ([1650 3].-mu)./sigma])\*theta; % You should change this

107. % ============================================================
109. fprintf(['Predicted price of a 1650 sq-ft, 3 br house ' ...
110. '(using gradient descent):\n $%f\n'], price);
112. fprintf('Program paused. Press enter to continue.\n');
113. pause;
115. %% ================ Part 3: Normal Equations ================
117. fprintf('Solving with normal equations...\n');
119. % ====================== YOUR CODE HERE ======================
120. % Instructions: The following code computes the closed form
121. % solution for linear regression using the normal
122. % equations. You should complete the code in
123. % normalEqn.m
124. %
125. % After doing so, you should complete this code
126. % to predict the price of a 1650 sq-ft, 3 br house.
127. %
129. %% Load Data
130. data = csvread('ex1data2.txt');
131. X = data(:, 1:2);
132. y = data(:, 3);
133. m = length(y);
135. % Add intercept term to X
136. X = [ones(m, 1) X];
138. % Calculate the parameters from the normal equation
139. theta = normalEqn(X, y);
141. % Display normal equation's result
142. fprintf('Theta computed from the normal equations: \n');
143. fprintf(' %f \n', theta);
144. fprintf('\n');

147. % Estimate the price of a 1650 sq-ft, 3 br house
148. % ====================== YOUR CODE HERE ======================
149. price = [1 1650 3] \* theta; % You should change this

152. % ============================================================
154. fprintf(['Predicted price of a 1650 sq-ft, 3 br house ' ...
155. '(using normal equations):\n $%f\n'], price);