**Programming Exercise 8:Anomaly Detection and Recommender Systems 第二部分**

2016年03月17日 20:29:36

大家好，我是Mac Jiang，今天和大家分享Coursera-Stanford University-Machine Learning-Programming Exercise 8:Anomaly Detection and Recommender Systems的第二部分Recommender Systems的实现过程，第一部分Anomaly Detection的实现过程的网址为：http://blog.csdn.net/a1015553840/article/details/50913824。虽然我的代码通过了系统的测试，但这不一定是最好的，如果博友发现错误或者有更好的实现思路请留言联系，谢谢！希望我的博客能给您的学习带来一些帮助！

这部分，吴恩达老师主要讲了协同过滤算法（collabrative filtering），先介绍一些符号的基本意思：

j:第j个用户，用来用户计数

i:第i部电影，用于电影计数

r(i,j):第j个用户对第i部电影的评价，r(i,j)=1表示第j个用户对第i部电影有评价，为0反之。

y(i,j):第j个用户对第i部电影的评分

n下标m：用户总数

n下标u：电影总数

theta(j)：用户j的参数

x(i)：电影i的特征

用户j对电影i的评价预测：theta(j)' \* x(i)

协同过滤的算法步骤是利用已有的用户对电影的评价，通过梯度下降算法（B-LFGS/共轭梯度等高级优化算法）求解最优电影特征矩阵X和用户信息矩阵Theta，对r(i,j)=0的对应点进行预测，同时利用这些信息可以得到用户的喜好，为用户推荐电影。注意，在进行学习算法之前需要进行均值归一化！

这次实验就是基于以上理论，对以上理论的具体实现。

**1.实验数据和文件说明**

数据集：

ex8\_movies.mat---用户对电影的评价信息，有两个矩阵，一个为评价矩阵R，另一个为R用来指示哪些位置有评价信息。

ex8\_movieParams.mat---X矩阵信息，Theta矩阵信息，用户人数num\_users，电影数num\_movies，电影特征数num\_features

movie\_ids.txt---用于存储电影名

文件：

ex8\_cofi.m---用于控制程序的进行过程

loadMovieList.m---将movie\_ids.txt的电影名导入到电影名数组中

computeNumericalGradient.m---用求导的方法计算梯度值

checkCostFuntion.m---确定我们自己用cifiCostFunc.m计算出来的梯度的正确性

fmincg.m---高级优化函数，通过迭代的方法寻找代价值J的局部最小值或全局最小值，并记录此时的参数Theta和参数X

normalizeRating.m---在训练之前对举证Y进行均值归一化

cofiCostFunc.m---计算代价函数J的值和此时对X的梯度值X\_grad以及对Theta的梯度值Theta\_grad，需要完善代码！

**2.ex8\_cofi.m的控制过程**

1. %% =============== Part 1: Loading movie ratings dataset ================
2. % You will start by loading the movie ratings dataset to understand the
3. % structure of the data.
4. %
5. fprintf('Loading movie ratings dataset.\n\n');
6. % Load data
7. load ('ex8\_movies.mat');
8. % Y is a 1682x943 matrix, containing ratings (1-5) of 1682 movies on
9. % 943 users
10. %
11. % R is a 1682x943 matrix, where R(i,j) = 1 if and only if user j gave a
12. % rating to movie i
13. % From the matrix, we can compute statistics like average rating.
14. fprintf('Average rating for movie 1 (Toy Story): %f / 5\n\n', ...
15. mean(Y(1, R(1, :))));
17. % We can "visualize" the ratings matrix by plotting it with imagesc
18. imagesc(Y);
19. ylabel('Movies');
20. xlabel('Users');
21. fprintf('\nProgram paused. Press enter to continue.\n');
22. pause;
23. %% ============ Part 2: Collaborative Filtering Cost Function ===========
24. % You will now implement the cost function for collaborative filtering.
25. % To help you debug your cost function, we have included set of weights
26. % that we trained on that. Specifically, you should complete the code in
27. % cofiCostFunc.m to return J.
29. % Load pre-trained weights (X, Theta, num\_users, num\_movies, num\_features)
30. load ('ex8\_movieParams.mat');
32. % Reduce the data set size so that this runs faster
33. num\_users = 4; num\_movies = 5; num\_features = 3;
34. X = X(1:num\_movies, 1:num\_features);
35. Theta = Theta(1:num\_users, 1:num\_features);
36. Y = Y(1:num\_movies, 1:num\_users);
37. R = R(1:num\_movies, 1:num\_users);
38. % Evaluate cost function
39. J = cofiCostFunc([X(:) ; Theta(:)], Y, R, num\_users, num\_movies, ...
40. num\_features, 0);
41. fprintf(['Cost at loaded parameters: %f '...
42. '\n(this value should be about 22.22)\n'], J);
43. fprintf('\nProgram paused. Press enter to continue.\n');
44. pause;
46. %% ============== Part 3: Collaborative Filtering Gradient ==============
47. % Once your cost function matches up with ours, you should now implement
48. % the collaborative filtering gradient function. Specifically, you should
49. % complete the code in cofiCostFunc.m to return the grad argument.
50. %
51. fprintf('\nChecking Gradients (without regularization) ... \n');
52. % Check gradients by running checkNNGradients
53. checkCostFunction;
54. fprintf('\nProgram paused. Press enter to continue.\n');
55. pause;
57. %% ========= Part 4: Collaborative Filtering Cost Regularization ========
58. % Now, you should implement regularization for the cost function for
59. % collaborative filtering. You can implement it by adding the cost of
60. % regularization to the original cost computation.
61. %
62. % Evaluate cost function
63. J = cofiCostFunc([X(:) ; Theta(:)], Y, R, num\_users, num\_movies, ...
64. num\_features, 1.5);
65. fprintf(['Cost at loaded parameters (lambda = 1.5): %f '...
66. '\n(this value should be about 31.34)\n'], J);
67. fprintf('\nProgram paused. Press enter to continue.\n');
68. pause;
70. %% ======= Part 5: Collaborative Filtering Gradient Regularization ======
71. % Once your cost matches up with ours, you should proceed to implement
72. % regularization for the gradient.
73. %
74. %
75. fprintf('\nChecking Gradients (with regularization) ... \n');
76. % Check gradients by running checkNNGradients
77. checkCostFunction(1.5);
78. fprintf('\nProgram paused. Press enter to continue.\n');
79. pause;
81. %% ============== Part 6: Entering ratings for a new user ===============
82. % Before we will train the collaborative filtering model, we will first
83. % add ratings that correspond to a new user that we just observed. This
84. % part of the code will also allow you to put in your own ratings for the
85. % movies in our dataset!
86. %
87. movieList = loadMovieList();
88. % Initialize my ratings
89. my\_ratings = zeros(1682, 1);
90. % Check the file movie\_idx.txt for id of each movie in our dataset
91. % For example, Toy Story (1995) has ID 1, so to rate it "4", you can set
92. my\_ratings(1) = 4;
93. % Or suppose did not enjoy Silence of the Lambs (1991), you can set
94. my\_ratings(98) = 2;
95. % We have selected a few movies we liked / did not like and the ratings we
96. % gave are as follows:
97. my\_ratings(7) = 3;
98. my\_ratings(12)= 5;
99. my\_ratings(54) = 4;
100. my\_ratings(64)= 5;
101. my\_ratings(66)= 3;
102. my\_ratings(69) = 5;
103. my\_ratings(183) = 4;
104. my\_ratings(226) = 5;
105. my\_ratings(355)= 5;
106. fprintf('\n\nNew user ratings:\n');
107. for i = 1:length(my\_ratings)
108. if my\_ratings(i) > 0
109. fprintf('Rated %d for %s\n', my\_ratings(i), ...
110. movieList{i});
111. end
112. end
114. fprintf('\nProgram paused. Press enter to continue.\n');
115. pause;
117. %% ================== Part 7: Learning Movie Ratings ====================
118. % Now, you will train the collaborative filtering model on a movie rating
119. % dataset of 1682 movies and 943 users
120. %
122. fprintf('\nTraining collaborative filtering...\n');
123. % Load data
124. load('ex8\_movies.mat');
125. % Y is a 1682x943 matrix, containing ratings (1-5) of 1682 movies by
126. % 943 users
127. %
128. % R is a 1682x943 matrix, where R(i,j) = 1 if and only if user j gave a
129. % rating to movie i
131. % Add our own ratings to the data matrix
132. Y = [my\_ratings Y];
133. R = [(my\_ratings ~= 0) R];
134. % Normalize Ratings
135. [Ynorm, Ymean] = normalizeRatings(Y, R);
136. % Useful Values
137. num\_users = size(Y, 2);
138. num\_movies = size(Y, 1);
139. num\_features = 10;
140. % Set Initial Parameters (Theta, X)
141. X = randn(num\_movies, num\_features);
142. Theta = randn(num\_users, num\_features);
143. initial\_parameters = [X(:); Theta(:)];
144. % Set options for fmincg
145. options = optimset('GradObj', 'on', 'MaxIter', 100);
146. % Set Regularization
147. lambda = 10;
148. theta = fmincg (@(t)(cofiCostFunc(t, Y, R, num\_users, num\_movies, ...
149. num\_features, lambda)), ...
150. initial\_parameters, options);
152. % Unfold the returned theta back into U and W
153. X = reshape(theta(1:num\_movies\*num\_features), num\_movies, num\_features);
154. Theta = reshape(theta(num\_movies\*num\_features+1:end), ...
155. num\_users, num\_features);
156. fprintf('Recommender system learning completed.\n');
157. fprintf('\nProgram paused. Press enter to continue.\n');
158. pause;
160. %% ================== Part 8: Recommendation for you ====================
161. % After training the model, you can now make recommendations by computing
162. % the predictions matrix.
163. %
164. p = X \* Theta';
165. my\_predictions = p(:,1) + Ymean;
167. movieList = loadMovieList();
169. [r, ix] = sort(my\_predictions, 'descend');
170. fprintf('\nTop recommendations for you:\n');
171. for i=1:10
172. j = ix(i);
173. fprintf('Predicting rating %.1f for movie %s\n', my\_predictions(j), ...
174. movieList{j});
175. end
177. fprintf('\n\nOriginal ratings provided:\n');
178. for i = 1:length(my\_ratings)
179. if my\_ratings(i) > 0
180. fprintf('Rated %d for %s\n', my\_ratings(i), ...
181. movieList{i});
182. end
183. end

Part1：Loading movie ratings dataset---导入用户对电影的评价数据并可视化

Part2：Colleborative Filtering Cost Function---这里取电影和评价的一个子集，用cofiCostFunc计算代价值J，以及梯度值X\_grad，Theta\_grad

Part3：Collaboretive Filtering Gradient---对上面子集用[f(x-epsilon)-f(x+epsilon)]/(2\*epsilon)的方法计算梯度，确定我们用举证方法求得梯度的正确性

Part4：Collaboretive Filtering Regularization---对算法进行正则化，改进cofiCostFunc.m的J加上正则化项

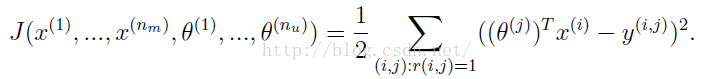
Part5：Collaboretive Flitering Gradient Rgulazation---对cofiCostFunc.m的求梯度过程进行正则化，对X\_grad,Theta\_grad加上正则化项

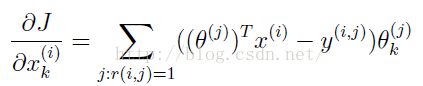
Part6：Entering ratings for a new user---新建一个用户，增加一些他的评价信息

Part7：Learning Movie Ratings---利用上面完善的协同过滤算法对数据进行学习

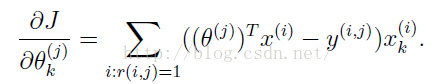
Part8：Recommendation for you---利用上步得到的学习参数，预测这位新用户可能喜欢的电影

**2.cofiCostFunc.m的实现过程**

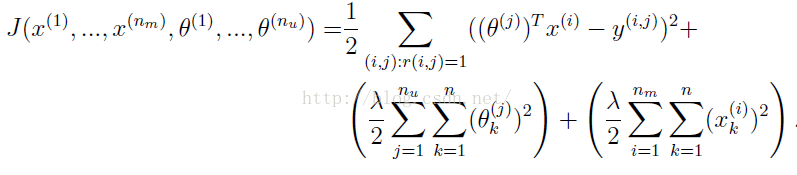
未正则化的代价函数值：J = sum(sum(((X \* Theta' - Y) .\* R).^2)) / 2;



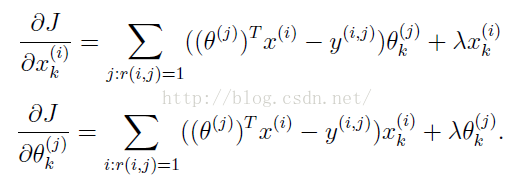
未正则化的X梯度值：X\_grad = (X \* Theta' - Y) .\* R \* Theta;



未正则化的Theta梯度值：Theta\_grad = (Theta \* X' - Y') .\* R' \* X;



正则化后的代价函数值：J = J + lambda / 2 \* (sum(sum(Theta .^2)) + sum(sum(X .^ 2)))



正则化后的X梯度值：X\_grad = X\_grad + lambda \* X;正则化后的Theta梯度值：Theta\_grad = Theta\_grad + lambda \* Theta;

由于推到过程比较复杂，而且推到过程在这里也不能打出来，所以就不说明了。不过大家可以从矩阵维度的角度寻找一些启发！具体实现代码如下：

1. function [J, grad] = cofiCostFunc(params, Y, R, num\_users, num\_movies, ...
2. num\_features, lambda)
3. %COFICOSTFUNC Collaborative filtering cost function
4. % [J, grad] = COFICOSTFUNC(params, Y, R, num\_users, num\_movies, ...
5. % num\_features, lambda) returns the cost and gradient for the
6. % collaborative filtering problem.
7. %
9. % Unfold the U and W matrices from params
10. X = reshape(params(1:num\_movies\*num\_features), num\_movies, num\_features);
11. Theta = reshape(params(num\_movies\*num\_features+1:end), ...
12. num\_users, num\_features);
13. % You need to return the following values correctly
14. J = 0;
15. X\_grad = zeros(size(X));
16. Theta\_grad = zeros(size(Theta));
18. % ====================== YOUR CODE HERE ======================
19. % Instructions: Compute the cost function and gradient for collaborative
20. % filtering. Concretely, you should first implement the cost
21. % function (without regularization) and make sure it is
22. % matches our costs. After that, you should implement the
23. % gradient and use the checkCostFunction routine to check
24. % that the gradient is correct. Finally, you should implement
25. % regularization.
26. %
27. % Notes: X - num\_movies x num\_features matrix of movie features
28. % Theta - num\_users x num\_features matrix of user features
29. % Y - num\_movies x num\_users matrix of user ratings of movies
30. % R - num\_movies x num\_users matrix, where R(i, j) = 1 if the
31. % i-th movie was rated by the j-th user
32. %
33. % You should set the following variables correctly:
34. %
35. % X\_grad - num\_movies x num\_features matrix, containing the
36. % partial derivatives w.r.t. to each element of X
37. % Theta\_grad - num\_users x num\_features matrix, containing the
38. % partial derivatives w.r.t. to each element of Theta
39. %
40. J = sum(sum(((X \* Theta' - Y) .\* R).^2)) / 2; %因为到多次调用高级优化函数，所以这里最好采用向量的方法，而不是利用for循环
41. X\_grad = (X \* Theta' - Y) .\* R \* Theta;%利用向量的方法计算对X的梯度
42. %利用向量的方法计算对Theta的梯度
43. J = J + lambda / 2 \* (sum(sum(Theta .^2)) + sum(sum(X .^ 2)));%对代价值进行正则化
44. X\_grad = X\_grad + lambda \* X;
45. Theta\_grad = Theta\_grad + lambda \* Theta;
46. % =============================================================
47. grad = [X\_grad(:); Theta\_grad(:)];
48. end

http://blog.csdn.net/a1015553840/article/details/50916801