第一步 高斯分布

```
function [mu sigma2] = estimateGaussian(X)
%ESTIMATEGAUSSIAN This function estimates the parameters of a
%Gaussian distribution using the data in X
%   [mu sigma2] = estimateGaussian(X),
%   The input X is the dataset with each n-dimensional data point in one row
%   The output is an n-dimensional vector mu, the mean of the data set
%   and the variances sigma^2, an n x 1 vector
%

% Useful variables
[m, n] = size(X);

% You should return these values correctly
mu = zeros(n, 1);
sigma2 = zeros(n, 1);

% ====================== YOUR CODE HERE ======================
% Instructions: Compute the mean of the data and the variances
%               In particular, mu(i) should contain the mean of
%               the data for the i-th feature and sigma2(i)
%               should contain variance of the i-th feature.
%
```

$$\mu_i = \frac{1}{m}\sum_{j=1}^{m} x_i^{(j)};$$

mu = (mean(X))'; 计算矩阵X每一列的平均值，然后转置，得到n*1维的向量miu

$$\sigma_i^2 = \frac{1}{m}\sum_{j=1}^{m}(x_i^{(j)} - \mu_i)^2$$

sigma2 = (mean((X-repmat(mean(X), m, 1)).^2))';

这个公式包含了以下五步：

1. repmat(mean(X), m, 1)将mean(X)的结果（1*n）复制m行，得到矩阵A（m*n）
2. 矩阵X减去矩阵A得到矩阵B（m*n）
3. 对矩阵B的每一项进行平方得到矩阵C（m*n）
4. 对矩阵C的每一列求平均值得到矩阵D（1*n）
5. 对矩阵D进行转置，得到sigma（n*1）

% ==

end

第二步 F1SCORE

The F_1 score is computed using precision ($prec$) and recall (rec):

$$F_1 = \frac{2 \cdot prec \cdot rec}{prec + rec}, \tag{3}$$

You compute precision and recall by:

$$prec = \frac{tp}{tp + fp} \tag{4}$$

$$rec = \frac{tp}{tp + fn}, \tag{5}$$

where

- tp is the number of true positives: the ground truth label says it's an anomaly and our algorithm correctly classified it as an anomaly.
- fp is the number of false positives: the ground truth label says it's not an anomaly, but our algorithm incorrectly classified it as an anomaly.
- fn is the number of false negatives: the ground truth label says it's an anomaly, but our algorithm incorrectly classified it as not being anomalous.

function [bestEpsilon bestF1] = selectThreshold(yval, pval)

%SELECTTHRESHOLD Find the best threshold (epsilon) to use for selecting

%outliers

% [bestEpsilon bestF1] = SELECTTHRESHOLD(yval, pval) finds the best

% threshold to use for selecting outliers based on the results from a

```
%   validation set (pval) and the ground truth (yval).
%

bestEpsilon = 0;
bestF1 = 0;
F1 = 0;

stepsize = (max(pval) - min(pval)) / 1000;
for epsilon = min(pval):stepsize:max(pval)

    % ====================== YOUR CODE HERE ======================
    % Instructions: Compute the F1 score of choosing epsilon as the
    %               threshold and place the value in F1. The code at the
    %               end of the loop will compare the F1 score for this
    %               choice of epsilon and set it to be the best epsilon if
    %               it is better than the current choice of epsilon.
    %
    % Note: You can use predictions = (pval < epsilon) to get a binary vector
    %       of 0's and 1's of the outlier predictions

predictions = (pval < epsilon); 满足条件的元素全部被标记为1

tp = sum(sum((predictions == 1 & yval == 1))); 用两次sum因为第一次会返回每一列符合条件的值，第二次才将行的数据加起来
fp = sum(sum((predictions == 1 & yval == 0)));
fn = sum(sum((predictions == 0 & yval == 1)));

prec = tp / (tp + fp);
rec = tp / (tp + fn);

F1 = (2 * prec * rec) / (prec + rec);

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

    if F1 > bestF1
       bestF1 = F1;
       bestEpsilon = epsilon;
    end
end
```

end

第三步 代价函数和梯度下降算法

```
function [J, grad] = cofiCostFunc(params, Y, R, num_users, num_movies, ...
                                  num_features, lambda)
%COFICOSTFUNC Collaborative filtering cost function
%   [J, grad] = COFICOSTFUNC(params, Y, R, num_users, num_movies, ...
%   num_features, lambda) returns the cost and gradient for the
%   collaborative filtering problem.
%

% Unfold the U and W matrices from params
X = reshape(params(1:num_movies*num_features), num_movies, num_features);
Theta = reshape(params(num_movies*num_features+1:end), ...
                num_users, num_features);



% You need to return the following values correctly
J = 0;
X_grad = zeros(size(X));
Theta_grad = zeros(size(Theta));

% ====================== YOUR CODE HERE ======================
% Instructions: Compute the cost function and gradient for collaborative
%               filtering. Concretely, you should first implement the cost
%               function (without regularization) and make sure it is
%               matches our costs. After that, you should implement the
%               gradient and use the checkCostFunction routine to check
%               that the gradient is correct. Finally, you should implement
%               regularization.
%
% Notes: X - num_movies  x num_features matrix of movie features
%        Theta - num_users  x num_features matrix of user features
%        Y - num_movies x num_users matrix of user ratings of movies
%        R - num_movies x num_users matrix, where R(i, j) = 1 if the
%            i-th movie was rated by the j-th user
%
% You should set the following variables correctly:
%
```

%   X_grad - num_movies x num_features matrix, containing the
%   partial derivatives w.r.t. to each element of X
%   Theta_grad - num_users x num_features matrix, containing the
%   partial derivatives w.r.t. to each element of Theta
%

% CostFunction

$$J(x^{(1)},...,x^{(n_m)},\theta^{(1)},...,\theta^{(n_u)}) = \frac{1}{2}\sum_{(i,j):r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \left(\frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}(\theta_k^{(j)})^2\right) + \left(\frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}(x_k^{(i)})^2\right)$$

allm = (X * Theta') - Y; 得出所有电影的预测评价和实际评价的差值
R = logical(R); 把R转化为逻辑向量
markedm = allm(R); 得到所有在R中被标记为1的元素（即所有被打过分的电影）
reg_one = (lambda/2) * sum(sum(Theta.^2)); 正规化1号
reg_two = (lambda/2) * sum(sum(X.^2)); 正规化2号

J = (1/2) * sum(markedm.^2) + reg_one + reg_two;

% Gradient

$$\frac{\partial J}{\partial x_k^{(i)}} = \sum_{j:r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})\theta_k^{(j)} + \lambda x_k^{(i)}$$

$$\frac{\partial J}{\partial \theta_k^{(j)}} = \sum_{i:r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(i)} + \lambda\theta_k^{(j)}.$$

X_grad = (((X * Theta') - Y) .* R) * Theta + lambda .* X;
Theta_grad = (((X * Theta') - Y) .* R)' * X + lambda .* Theta;
注意不要漏掉红色标记的地方，即把没有被打分的电影的参数归0

% ===

grad = [X_grad(:); Theta_grad(:)];

end