ex8 异常检测与推荐系统

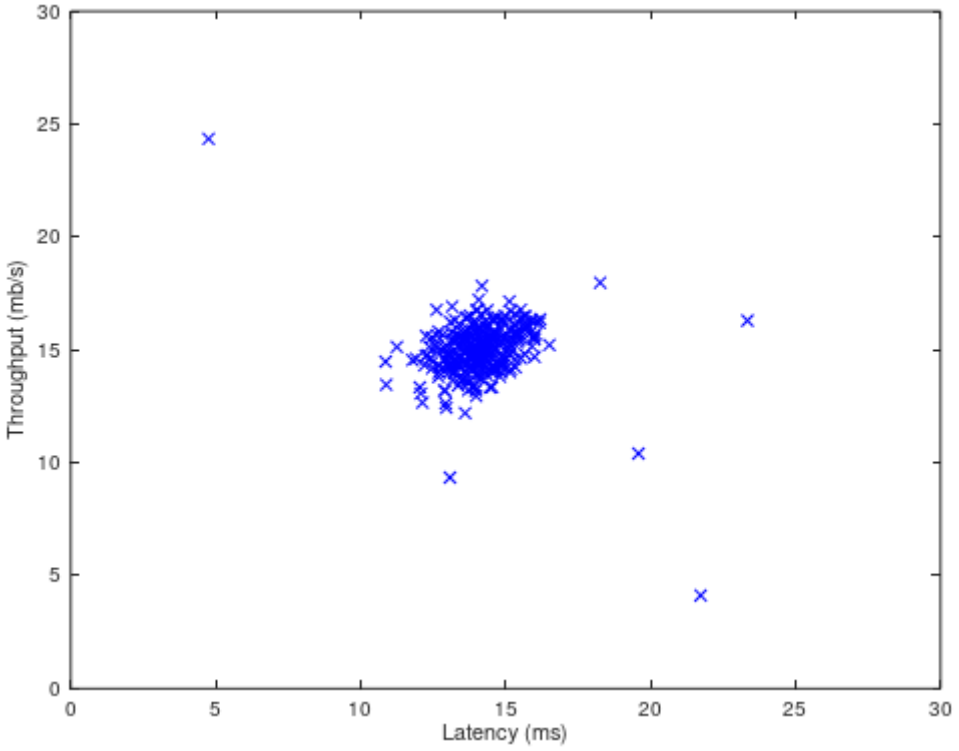
介绍：

本练习中需要实现异常检测算法并将其应用于检测网络中故障的服务器，在第二部分练习中使用协同过滤为电影创建推荐系统。

第一部分 异常检测

在本练习中实现异常检测算法来检测服务器计算机中的异常表现，特征为服务器响应的流量（mb/s）和延迟（ms）。在你的服务器正在运行的时候，收集它们表现的m=307个样本，由此获得一个无标签数据集{x (1) ,...,x (m) }，我们猜测这些样本的大多数是服务器正常运行的无异常样本，但在数据集中也存在一些代表服务器运行异常的样本。

我们将会使用高斯模型来检测数据集中异常的样本，首先从一个2维数据集开始，对该算法的运行进行可视化，在那个数据集上将会拟合出一个高斯分布，找到概率很低的值并将其认定为是异常值。之后，你将应用这个异常检测算法到一个拥有很多维度的更大的数据集上。运行ex8.m完成这部分的练习。



1.1 高斯分布

为了执行异常检测，我们首先需要做的是对数据的分布拟合出一个模型，给你一个训练集 {x (1) ,...,x (m) }，每个样本均有n个特征值，需要对每一个特征评

估出一个高斯分布，对于每个特征i=1...n，都需要找到能拟合数据（训练集的第i个维度）的 μ i 和σ 2。

高斯公式如下，μ为均值，σ 2控制方差：

$$p(x;\mu,\sigma^2)=\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

1.2 高斯分布的参数估计

你可以通过以下公式估计第i个特征的参数 (μ i , σ 2i)，估计均值，使用：

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

获取方差，使用：

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

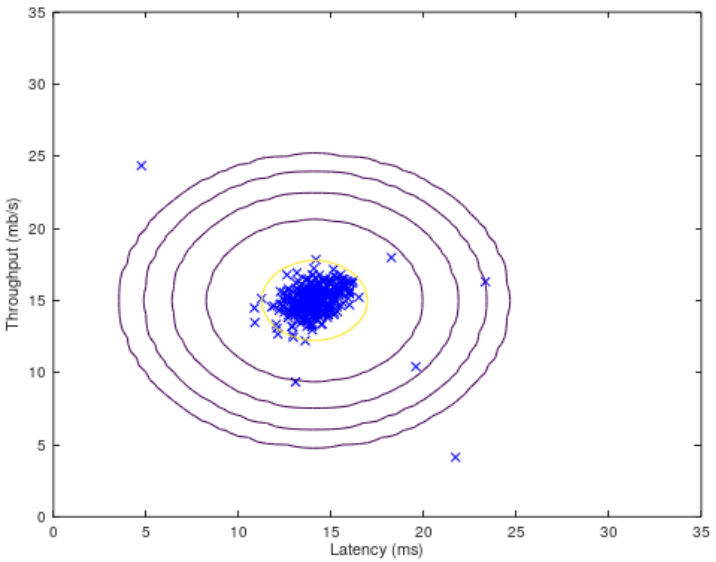
我们的任务是完成estimateGaussian.m文件，该函数以数据矩阵X为输入，以n维向量mu为输出，包含n个特征的均值，另一个n维向量sigma2包含所有特征的方差。可以通过对每个特征和每个样本使用一个for循环来实现上述函数（向量化的实现方式可以更加高效，值得一试），注意在Octave中var函数在计算 σ 2i的时候默认使用1/(m-1)，而不是使用1/m。

在完成estimateGaussian.m后，运行ex8.m文件后可以对拟合后的高斯分布的等高线进行可视化，我们可以观察到大多数的样本均处于最高概率的区域范围内，而异常样本则处于较低概率的区域范围内。

代码：

```
mu=sum(X,1)/m;
sigma2=sum((X-mu) .^2,1)/m;
```

运行结果：



第一次提交：

```
                        Part Name |     Score | Feedback
                        --------- |     ----- | --------
   Estimate Gaussian Parameters |  15 /  15 | Nice work!
                 Select Threshold |   0 /  15 |
     Collaborative Filtering Cost |   0 /  20 |
Collaborative Filtering Gradient |   0 /  30 |
                 Regularized Cost |   0 /  10 |
             Regularized Gradient |   0 /  10 |
                        ---------------------------------
                                  |  15 / 100 |
```

1.3 选择阈值， ε

完成高斯分布的参数估计之后，我们就可以研究研究哪些样本在这个分布上拥有较高的概率，哪些只有较低的概率。低概率的样本更有可能是我们的数据集中的异常样本。基于交叉验证集选择阈值来决定哪些样本属于异常样本是一种好的方法。在这部分练习中，你将实现一个算法，其通过在交叉验证集上计算F score来选择阈值 ε。

现在完成selectThreshold.m文件，我们将会使用到一个交叉验证集。带标签1表明其为异常样本，标签为0则表明为正常样本。对于交叉验证集的样本，我们都将计算其出现概率。这些概率的向量被放置到pval变量中，作为参数传入selectThreshold.m函数中，相关标签组成的向量也被放入yval变量中进行传参。

selectThreshold.m函数中需要返回两个值，一个是最终选择的阈值 ε，如果一个样本的概率小于这个值，该样本将会被认定为是异常值。函数也会返回一个值F1 score，其表明在给定阈值的情况下，找到ground truth异常的效果怎么样。对于不同的 ε值，通过当前阈值正确/错误分辨交叉验证集样本的情况计算F1 score。

F1有准确率和召回率求得：

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

准确率和召回率可以通过以下公式求得：

$$prec = \frac{tp}{tp+fp}$$
$$rec = \frac{tp}{tp+fn}$$

注：

tp是true positive的数量：真实标签表明异常而预测也判定为异常

fp是false positive的数量：真实标签表明不是异常而判定是异常

fn是false negative的数量：真实标签表明是异常而判定不是异常

在selectThreshold.m函数中，已经有一个尝试不同 ε的循环loop，并会基于最终的F1 score进行 ε的选择。完成selectThreshold.m函数，通过对所有交叉验证集的样本使用循环（计算tp、fp、fn）来求得F1 score。期望得到一个epsilon为8.99e-05。

注意：为了计算tp、fp、fn，我们可以使用 fp = sum((cvPredictions == 1) &(yval == 0))

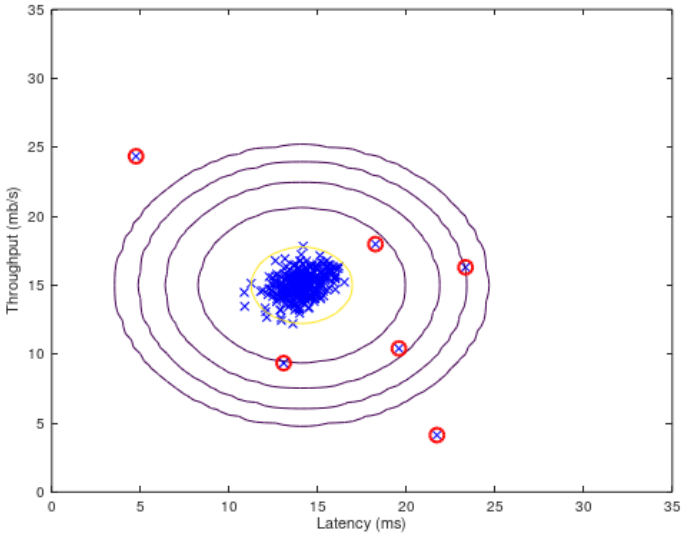
完成selectThreshold.m代码后，运行ex8.m可以将异常点标注出来。

代码：

```
predictions = (pval < epsilon);
tp=sum((predictions==1)&(yval==1));
fp=sum((predictions==0)&(yval==1));
fn=sum((predictions==1)&(yval==0));
prec=tp/(tp+fp);
rec=tp/(tp+fn);
F1=2*prec*rec/(prec+rec);
```

运行结果：

```
Best epsilon found using cross-validation: 8.990853e-05
Best F1 on Cross Validation Set:  0.875000
   (you should see a value epsilon of about 8.99e-05)
```



第二次提交：

```
                        Part Name |     Score | Feedback
                        --------- |     ----- | --------
   Estimate Gaussian Parameters |  15 /  15 | Nice work!
                 Select Threshold |  15 /  15 | Nice work!
     Collaborative Filtering Cost |   0 /  20 |
 Collaborative Filtering Gradient |   0 /  30 |
                Regularized Cost |   0 /  10 |
            Regularized Gradient |   0 /  10 |
                        ---------------------------------
                                 |  30 / 100 |
```

1.4 高维度的数据集

ex8.m的最后一部分会对更真实、更困难的数据集运行异常检测算法，每个样本包含从你的计算机服务器中抓取的更多的属性（11个特征），该脚本将会用你的代码对高斯分布进行参数估计，评估训练集X和交叉验证集Xval在该分布上的出现概率，最终通过selectThreshold.m函数找到最佳的阈值 ε，我们将会得到一个阈值为1.38e-18,查找出117个异常点。

```
Best epsilon found using cross-validation: 1.377229e-18
Best F1 on Cross Validation Set:  0.615385
# Outliers found: 117
```

第二部分 推荐系统

该部分练习中将要实现一个协同过滤算法并将其应用于一个电影评分的数据集，评分数值为1-5，该数据集有nu=943个使用者，有nm=1682部影片，你将要在该部分的练习中使用ex8_cofi.m文件。在更后面的练习中，你需要实现cofiCostFunc.m函数，用来计算协同过滤的目标函数与梯度，实现损失函数与梯度的代码后，使用fmincg.m函数来求得协同过滤的参数。

2.1 电影评分数据集

ex8_cofi.m脚本的第一部分将会加载 ex8_movies.mat数据集,在我们的Octave环境中提供Y变量和R变量。

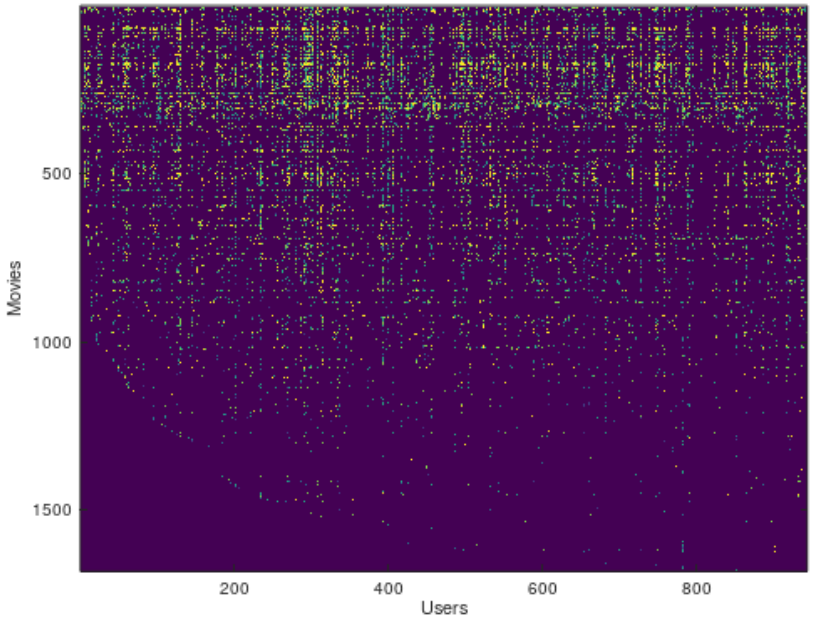
Y矩阵（num_users*num_movies）存储评分数值y(i,j)（1-5），R矩阵是一个二值指标矩阵，如果某个用户对某一部电影进行了打分，则对应该矩阵中的元素为1，否则即为0。协同过滤的目标是为了对用户尚未进行打分的每部电影进行评分的预测，R(i,j) = 0，这能使我们将预测打分最高的电影推荐给用户。

为了方便理解Y矩阵，ex8_cofi.m将会计算第一部电影ToyStory的评分的均值，并将平均分数输出到屏幕上。本练习中也会用到X矩阵和Theta矩阵：

$$\mathtt{X} = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — (x^{(n_m)})^T — \end{bmatrix}, \quad \mathtt{Theta} = \begin{bmatrix} — (\theta^{(1)})^T — \\ — (\theta^{(2)})^T — \\ \vdots \\ — (\theta^{(n_u)})^T — \end{bmatrix}$$

X矩阵的第i行表示第i部电影的特征向量，Theta的第j行表示第j个用户的参数向量。上述两个向量均是n维向量。我们令n=100，则上述两个向量均为100维，相应地，X为

nm*100维度的矩阵，Theta为nu*100维度的矩阵。




```
Average rating for movie 1 (Toy Story): 3.878319 / 5
```

2.2 协同过滤算法

先实现不带正则项的损失函数，协同过滤算法需要考虑一组参数向量x(1)...x(nm)，θ(1)...θ(nu)，建立的模型通过

$$y^{(i,j)} = (\theta^{(j)})^T x^{(i)}$$

来预测每个用户对于某部电影的评分，给你一份数据集，包含某些用户对某些电影的评分，期望学习出完成最佳拟合的参数向量，即获得最小的平方误差。完成cofiCostFunc.m文件来为协同滤波计算损失函数与梯度，函数的参数为X和Theta，为了使用现成的最小化器如fmincg损失函数将参数展开为向量params，我们以前在神经网络的练习中使用过同样的向量展开方法。

2.2.1 协同过滤的损失函数

不带正则项的损失函数：

$$J(x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)}) = \frac{1}{2} \sum_{(i,j):r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2$$

调整 cofiCostFunc.m函数来返回损失值J，注意只有在R(i,j)=1的时候才会积累用户j和电影i的损失值。运行ex8_cofi.m文件将会运行你的损失函数。

注意：我们强烈建议使用向量化的方法实现损失值J的计算，因为它将会在后续被优化包fmincg调用很多次，虽然非向量化的实现方法是最简单的。为了进行向量化的实现，有以下提示：可以通过R矩阵将被选中的条目设置为0，比如R *M表示元素之间进行相乘。由于R矩阵只含有值为0或1的元素，因此只有当某个R矩阵中的元素为0的时候才能将M矩阵中

对应的元素设置为0，因此sum(sum(R.*M))是M中与R中值为1的元素对应的所有元素的值的和。

代码：

J=sum(sum(((X*Theta'-Y) .^2) .*R))/2;

运行结果：

```
Cost at loaded parameters: 22.224604
(this value should be about 22.22)
```

第三次提交：

```
                        Part Name |     Score | Feedback
                        --------- |     ----- | --------
    Estimate Gaussian Parameters |  15 /  15 | Nice work!
                  Select Threshold |  15 /  15 | Nice work!
      Collaborative Filtering Cost |  20 /  20 | Nice work!
Collaborative Filtering Gradient |   0 /  30 |
                  Regularized Cost |   0 /  10 |
              Regularized Gradient |   0 /  10 |
                        ---------------------------------
                                   |  50 / 100 |
```

2.2.2 协同过滤的梯度

现在实现不带正则项的梯度计算，继续完成cofiCostFunc.m中的代码，返回X_grad和Theta_grad变量的值，注意X_grad应该是和X矩阵同样大小的矩阵，Theta_grad也适合Theta一样大小的矩阵，损失函数的梯度如下所示：

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

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

注意：该函数返回的两组变量的梯度被展开成一个向量，在完成计算梯度的代码后，ex8_cofi.m将会运行checkCostFunction函数来从数学上检验你实现的梯度计算的函数，如果实现是正确的，你会发现解析梯度和数值梯度大小是匹配的。

实现时注意：如果不使用向量化实现，训练将会变得慢得多（几个小时），强烈建议使用向量化实现，刚开始的时候可以通过对电影和用户使用一个for循环来实现梯度计算，完成这一部分后可以使用一个非向量化的版本，通过实现另一个内部for循环来计算求和中的每个元素，在用这种方式完成后，尝试使用向量化实现（对内部for循环进行向量化），这样就只剩下两个for循环了。

实现Tips：为了执行向量化的操作，你可以这样做：我们需要有一种方法同时计算每一个特征向量的全部偏微分项，进行如下定义：

$$(\mathtt{X_{grad}(i,:)})^T = \begin{bmatrix} \frac{\partial J}{\partial x_1^{(i)}} \\ \frac{\partial J}{\partial x_2^{(i)}} \\ \vdots \\ \frac{\partial J}{\partial x_n^{(i)}} \end{bmatrix} = \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})\theta^{(j)}$$

为了向量化上述表达式，从Theta和Y开始，只选择感兴趣元素（用户进行过打分的电影）的索引，当你考虑第i部电影的特征时，只需要注意对这部电影打过分的用户，这将使你将其余用户从Theta和Y矩阵中移除出去。

具体地， idx = find(R(i, :)==1)可以获得用户进行过打分的电影的索引列表，产生临时矩阵Theta temp = Theta(idx,:)以及 Y temp = Y(i,idx)，这样偏微分计算就可以是如下形式（返回行向量）：

$$\mathtt{X_{grad}(i,:)} = (\mathtt{X(i,:)} * \mathtt{Theta_{temp}^{T}} - \mathtt{Y_{temp}}) * \mathtt{Theta_{temp}}$$

对x(i)的偏微分进行向量化求导之后，也可以用同样的方式对θ(j)进行向量化求导计算。

代码：

(1)非向量化：

```
for i=1:size(X,1)
  idx1=find(R(i,:)==1);
  ThetaTemp=Theta(idx1,:);
  YTemp1=Y(i,idx1);
  X_grad(i,:)=(X(i,:)*ThetaTemp'-YTemp1)*ThetaTemp;
endfor
for j=1:size(Theta,1)
  idx2=find(R(:,j)==1);
  XTemp=X(idx2,:);
  YTemp2=Y(idx2,j)';
  Theta_grad(j,:)=(Theta(j,:)*XTemp'-YTemp2)*XTemp;
endfor
```

（2）向量化：

```
X_grad=((X*Theta'-Y) .*R)*Theta;
Theta_grad=((X*Theta'-Y) .*R)'*X;
```

运行结果：

```
Checking Gradients (without regularization) ...
   -1.80252   -1.80252
   -2.51344   -2.51344
    2.42382    2.42382
   -0.51078   -0.51078
    0.89564    0.89564
    0.08166    0.08166
   -2.88203   -2.88203
   -0.47546   -0.47546
    5.27582    5.27582
    8.32381    8.32381
    8.40027    8.40027
    2.19664    2.19664
   -3.18325   -3.18325
    2.68911    2.68911
   -3.74583   -3.74583
  -12.77620  -12.77620
    0.00000    0.00000
    2.01427    2.01427
   -0.18489   -0.18489
    0.15124    0.15124
   -0.73746   -0.73746
    0.00000    0.00000
   -4.41313   -4.41313
   -1.10797   -1.10797
   -7.63595   -7.63595
   -4.86778   -4.86778
    0.00000    0.00000
The above two columns you get should be very similar.
(Left-Your Numerical Gradient, Right-Analytical Gradient)

If your backpropagation implementation is correct, then
the relative difference will be small (less than 1e-9).

Relative Difference: 1.17592e-12
```

第四次提交：

```
                      Part Name |     Score | Feedback
                      --------- |     ----- | --------
    Estimate Gaussian Parameters |  15 /  15 | Nice work!
                 Select Threshold |  15 /  15 | Nice work!
    Collaborative Filtering Cost |  20 /  20 | Nice work!
Collaborative Filtering Gradient |  30 /  30 | Nice work!
                  Regularized Cost |   0 /  10 |
              Regularized Gradient |   0 /  10 |
                      ---------------------------------
                                   |  80 / 100 |
```

2.2.3 损失函数正则化

公式如下：

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

现在需要将正则项加入到原始的损失函数计算J中，运行ex8_cofi.m将会计算损失值，期望得到的损失值为31.34。

代码：

```
J=sum(sum(((X*Theta'-Y) .^2) .*R))/2+lambda*(sum(sum(X .^2))+sum(sum(Theta .^2)))/2;
```

运行结果：

```
Cost at loaded parameters (lambda = 1.5): 31.344056
(this value should be about 31.34)
```

第五次提交：

```
                  Part Name |     Score | Feedback
                  --------- |     ----- | --------
   Estimate Gaussian Parameters |  15 /  15 | Nice work!
                Select Threshold |  15 /  15 | Nice work!
    Collaborative Filtering Cost |  20 /  20 | Nice work!
Collaborative Filtering Gradient |  30 /  30 | Nice work!
                Regularized Cost |  10 /  10 | Nice work!
            Regularized Gradient |   0 /  10 |
                  ---------------------------------
                                   |  90 / 100 |
```

2.2.4 梯度正则化

通过加入正则项的贡献来使cofiCostFunction.m返回正则化梯度，公式如下：

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

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

这意味着只需要为前面声明的变量 X_grad(i,:)加入λx(i)，为Theta grad(j,:)加入λθ(j)。完成计算梯度值的代码后，ex8_cofi.m将会调用checkCostFunction来检测梯度计算实现的正确性。

代码：

```
X_grad=((X*Theta'-Y) .*R)*Theta+lambda*X;
Theta_grad=((X*Theta'-Y) .*R)'*X+lambda*Theta;
```

运行结果：

```
Checking Gradients (with regularization) ...
  -1.10209  -1.10209
   1.53999   1.53999
  -1.36191  -1.36191
   6.11731   6.11731
  -5.06309  -5.06309
   3.12290   3.12290
   1.84195   1.84195
   5.04834   5.04834
   8.72324   8.72324
   3.00982   3.00982
  -1.65620  -1.65620
   5.90327   5.90327
   0.69834   0.69834
  -4.09017  -4.09017
  -3.39815  -3.39815
   3.51974   3.51974
  -2.20803  -2.20803
  -0.11660  -0.11660
  -3.43012  -3.43012
  -0.31884  -0.31884
   3.11692   3.11692
  -3.57258  -3.57258
  -1.38799  -1.38799
  -2.32694  -2.32694
  -4.52804  -4.52804
   5.37850   5.37850
   7.96368   7.96368
The above two columns you get should be very similar.
(Left-Your Numerical Gradient, Right-Analytical Gradient)

If your backpropagation implementation is correct, then
the relative difference will be small (less than 1e-9).

Relative Difference: 2.29352e-12
```

第六次提交：

```
                        Part Name |     Score | Feedback
                        --------- |     ----- | --------
    Estimate Gaussian Parameters |  15 /  15 | Nice work!
                  Select Threshold |  15 /  15 | Nice work!
      Collaborative Filtering Cost |  20 /  20 | Nice work!
  Collaborative Filtering Gradient |  30 /  30 | Nice work!
                  Regularized Cost |  10 /  10 | Nice work!
              Regularized Gradient |  10 /  10 | Nice work!
                                   ----------------------------------
                                   | 100 / 100 |
```

2.3 电影推荐系统的学习

完成协同过滤的损失函数与梯度计算后，可以开始用算法为自己训练电影推荐系统，ex8_cofi.m的下一部分，你将会研究自己的电影喜好，这样在后面运行算法的时候你就可以获得属于自己的电影推荐。我们根据自己的喜好填写了一些数据，但是你应该基于自己的喜好进行改变，关于所有电影及其打分的数据集能在 movie_idx.txt文件中找到。

2.3.1 推荐

在另外的评分被加入数据集之后，脚本将会继续训练协同过滤模型，这将获得参数X和Theta，通过：

$$y^{(i,j)} = (\theta^{(j)})^T x^{(i)}$$

就可以对某个用户对某部电影的评分进行预测，程序的下一部分将会计算出所有这样的评分并将推荐的电影进行显示（基于传递进去的初始化评分参数），注意随着随机初始值的不同，我们获得的预测值也会不一样。

最终效果：

```
Training collaborative filtering...
Iteration   100 | Cost: 3.895241e+04
Recommender system learning completed.
```

```
Top recommendations for you:
Predicting rating 5.0 for movie Someone Else's America (1995)
Predicting rating 5.0 for movie Entertaining Angels: The Dorothy Day Story (1996)
Predicting rating 5.0 for movie They Made Me a Criminal (1939)
Predicting rating 5.0 for movie Marlene Dietrich: Shadow and Light (1996)
Predicting rating 5.0 for movie Star Kid (1997)
Predicting rating 5.0 for movie Saint of Fort Washington, The (1993)
Predicting rating 5.0 for movie Prefontaine (1997)
Predicting rating 5.0 for movie Santa with Muscles (1996)
Predicting rating 5.0 for movie Aiqing wansui (1994)
Predicting rating 5.0 for movie Great Day in Harlem, A (1994)
```

```
Original ratings provided:
Rated 4 for Toy Story (1995)
Rated 3 for Twelve Monkeys (1995)
Rated 5 for Usual Suspects, The (1995)
Rated 4 for Outbreak (1995)
Rated 5 for Shawshank Redemption, The (1994)
Rated 3 for While You Were Sleeping (1995)
Rated 5 for Forrest Gump (1994)
Rated 2 for Silence of the Lambs, The (1991)
Rated 4 for Alien (1979)
Rated 5 for Die Hard 2 (1990)
Rated 5 for Sphere (1998)
```