统计信号处理大作业

最小二乘法

姓名：王道烩
学号：2015011006
班级：无 52

一、实验目的

使用最小二乘法解决协同滤波问题。

二、实验原理

要填充的矩阵可以表示为：$\mathrm{M}=UV^T$,其中$M_{ij}=U(i,:)V(j,:)^T$，表示用户 i 对电影 j 的打分是用户 i 的隐变量$\mathrm{U}(i,:)$与电影 j 的隐变量$\mathrm{V}(j,:)$的内积。考虑到用户类型和电影类型有限，因此有理由相信评分矩阵 M 是低秩的。由于矩阵的秩是矩阵奇异值向量的 0 范数，可以将其放缩到 1 范数，即矩阵的核范数。由于举证的核范数满足：

$$||\mathbf{X}||_*=\min_{\mathbf{U},\mathbf{V}|\mathbf{X}=\mathbf{U}\mathbf{V}^T}\frac{1}{2}(||\mathbf{U}||_F^2+||\mathbf{V}||_F^2)$$

因此上述问题可以通过优化下述目标函数进行求解：

$$\min_{\mathbf{U},\mathbf{V}}||\mathbf{W}*(\mathbf{M}-\mathbf{U}\mathbf{V}^T)||_F^2+\frac{\lambda}{2}(||\mathbf{U}||_F^2+||\mathbf{V}||_F^2)$$

其中λ为控制矩阵低秩程度的超参数。W 为标志矩阵，$\mathrm{W}(i,j)$=1 代表用户 i 对电影 j 已经打过分了，$\mathrm{W}(i,j)$=0 表示未打分，*表示矩阵对应元素相乘。

对于上述优化问题一般有两种思路，本人采用的方法为交替最小二乘法，及固定 U 以 V 作为优化变量，固定 V 以 U 作为优化变量，交替地进行此过程的求解。

公式推导如下：（其中*代表数量乘 .*代表矩阵相乘 U(i,:) 为 U 的第 i 行 V(:,i)为 V 的第 i 列）

（1）固定 V 以 U 作为变量时：

$$\frac{\partial \mathrm{L}}{\partial \mathrm{U}_{\mathrm{ik}}}=\sum_{j=1}^{n}2(M_{ij}-X_{ij})(-V_{ij})W_{ij}+\lambda U_{\mathrm{ij}}$$

$$\therefore \frac{\partial \mathrm{L}}{\partial \mathrm{U}(i,:)}=2[W(i,:)*M(i,:)-W(i,:)*X(i,:)].*(-V)+\lambda U(i,:)=0$$

$$\lambda \mathrm{U}(i,:)=2[W(i,:)*M(i,:)-W(i,:)*X(i,:)].*V$$
$$=2[W(i,:)*M(i,:)].*V-2[W(i,:)*X(i,:)].*V$$

又因为

$$\mathrm{X}(i,:)=U(i,:).*V^T$$
$$W(i,:)*X(i,:)=W(i:)*[U(i,:).*V^T]$$
$$=U(i,:).*[W(i,:)*V^T]$$

所以上式可写为：

$$\lambda U(i,:) = 2[W(i,:)*M(i,:)].*V - 2U(i,:).*[W(i,:)*V^T].*V$$

$$U(i,:).*\{\lambda I + 2[W(i,:)*V^T].*V\} = 2[W(i,:)*M(i,:)].*V$$

$$U(i,:) = 2[W(i,:)*M(i,:)].*V.*\{\lambda I + 2[W(i,:)*V^T].*V\}^{-1}$$

这样就得到了每次 U 一行的更新公式

（2）固定 U 以 V 为变量时：

$$\frac{\partial L}{\partial V_{ik}} = \sum_{j=1}^{M} 2(M_{ji} - X_{ji})(-U_{jk})W_{jk} + \lambda V_{ik}$$

$$\frac{\partial L}{\partial V(i,:)} = 2[M(:,i)*W(:,i) - X(:,i)*W(:,i)]^T.*(-U) + \lambda V(i,:)$$

$$= 2[M^T(i,:)*W^T(i,:) - X^T(i,:)*W^T(i,:)].*(-U) + \lambda V(i,:) = 0$$

$$\lambda V(i,:) = 2[W^T(i,:)*M^T(i,:) - W^T * X^T(i,:)].*U$$

又因为

$$X^T = V.*U^T$$

$$X^T(i,:) = V(i,:).*U^T$$

$$W^T(i,:)*[V(i,:).*U^T] = V(i,:).*[W^T(i,:)*U^T]$$

所以上式可以化简如下：

$$V(i,:),*\{\lambda I + 2[W^T(i,:)*U^T].*U\} = 2[W^T(i,:)*M^T(i,:)].*U$$

$$V(i,:) = 2[W^T(i,:)*M^T(i,:)].*U.*\{\lambda I + 2[W^T(i,:)*U^T].*U\}^{-1}$$

上式就时 V 每一每一行的更新公式。

将上面的更新公式用代码表示如下：

```
temple = np.linalg.inv(np.eye(feature_size) * lamda + 2 * (W[i , :] * V.T).dot(V))
U[i , :] = 2 * ((W[i , :] * M[i , :]).dot(V)).dot(temple)
```

```
temple = np.linalg.inv(np.eye(feature_size) * lamda +2* ((W.T)[i , :] * U.T).dot(U))
V[i , :] =2 * (((W.T)[i ,:] * (M.T)[i ,:]).dot(U)).dot(temple)
```

三、实验内容

（1）实验环境

操作系统：Ubuntu16.04LTS

工具：python3.5 + numpy + scipy + matplotlib

硬件：CPU：8*i7 7700HQ　　内存：8G

（2）实验步骤

本次实验一共主要有有三个文件，其中：

select_data.py 文件用于挑选训练集和测试集。一共实现了三种挑选方法：随机法，均匀挑选测试集和均匀挑选训练集。通过适当的添加删除注释可以选择某种方法来生成文件 data_set_my.mat 文件。内含训练集和测试集数据。

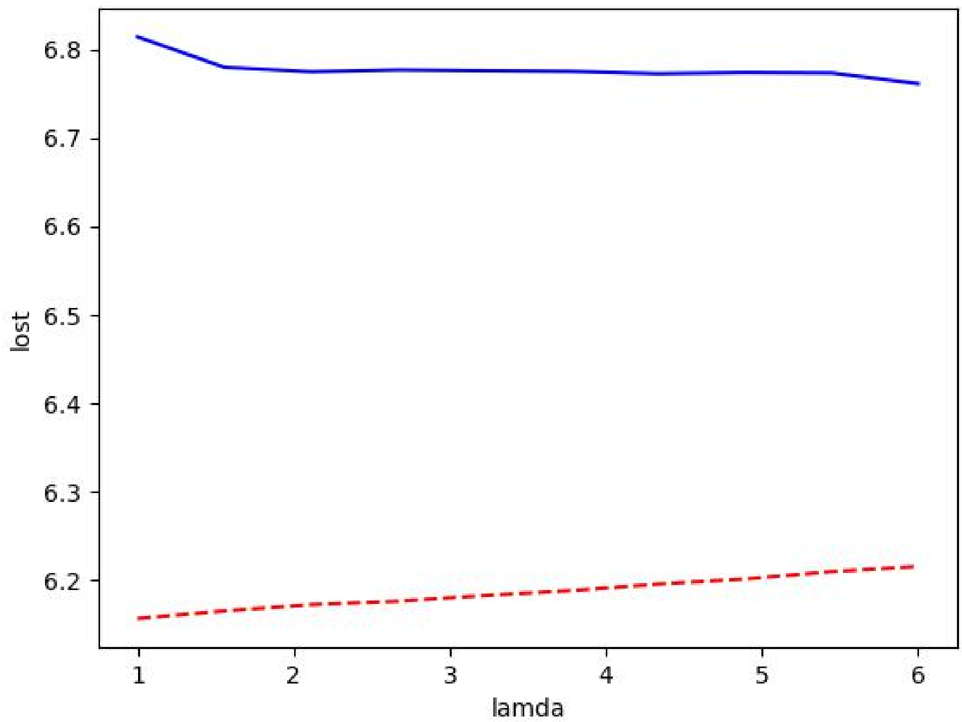
ALS.py 文件是算法主题，实现了交替最小二乘算法，通过这行次文件可以读取 data_set_my.mat 文件，并将得到的 U V result= $U.*V^T$ 保存在 resulr.mat 文件中。

Plot.py 文件主要用于画图来分析损失随着参数λ以及特征维度 feature_size 的变化趋势，从而确定最终选择的参数。

四、实验结果分析

（1）参数λ的影响

首先，我选择一个固定的训练集和测试集，来通过选择不同的λ来观察在训练集上的损失和在测试集上的损失，得到的图象如下：

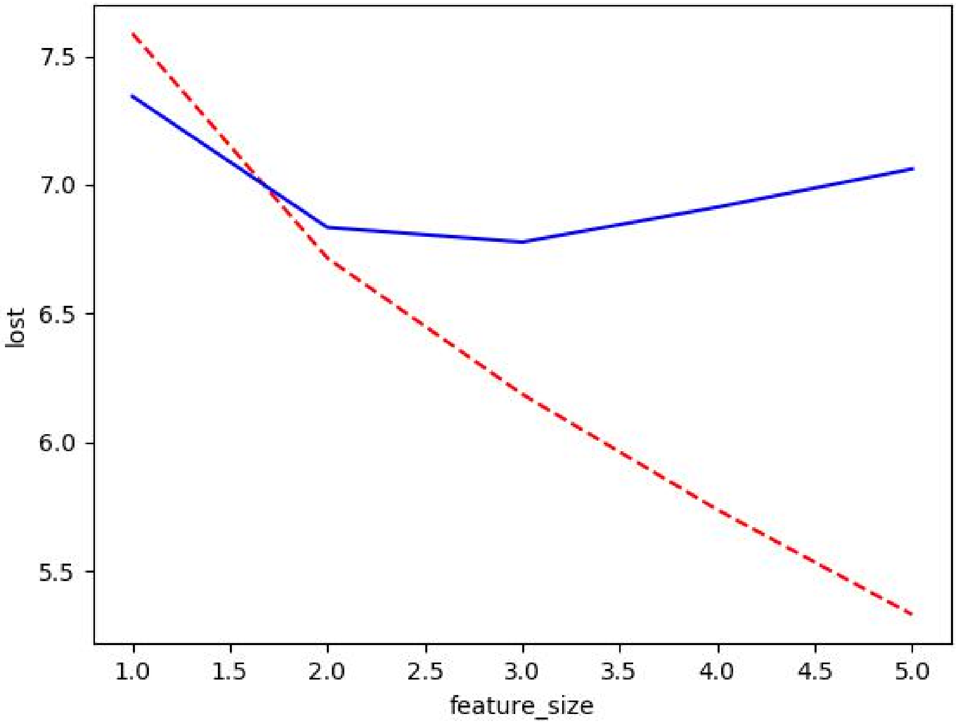


其中红色虚线为在训练集上的损失，蓝色实现为在测试集上的损失。

可以观察到，随着λ的增大，在训练集上的损失会增加而在测试集上的损失会减小，这是因为λ的作用可以看做一种避免过拟合的效果。通过增加λ的值，能够使我们得到的模型更加平滑，这会增加在训练集上的损失，但是增加了模型的泛化能力，所以其在训练集上的损失会降低。通过不断的实验，最终将参数λ选为 10。

（2）特征维度 feature_size 的影响

通过固定λ，改变 feature_size，可以得到不同的特征维度，在训练集和测试集上的损失变化如下：



其中红色虚线为在训练集上的损失，蓝色实线为在测试集上的损失。可以看到，随着特征维度的增加，在训练集上的损失是逐渐降低的，但是在测试集上的损失是现降低再上升的。这是因为，特征维度越大，模型的代表能力就越强，就越能够精确地拟合训练数据，使得在训练集上的损失降低，但是，当特征维度超过某一个值之后，就会出现过拟合现象，这使得模型的泛化能力下降，从而使得其在测试集上的损失上升。

（3）训练集以及测试集选取方法的对比

本人在选择训练集和测试集方面一共使用了三种方法，下对这三种方法进行分析：

随机法：这种方法是最简单的方法，在提供的数据集上随机选取 10000 个作为测试集，剩下的作为训练集，这种方法得到的最终在测试集上的损失大致为 7.8 左右。效果不是非常理想。

均匀选取测试集：通过计数，本人发现每行的数据个数最小为 15，我在每行上面交替选取 11 个和 10 个数据作为测试集，剩下的作为训练集，这样得到的在测试集上面的损失大致为 8.1 左右，效果更差。

均匀选取训练集：通过两次循环，保证选取的数据集在行和列上分布比较均匀。这样最终得到的在测试集上的损失大致在 7.0 左右，效果比较好。

通过以上对比，本人最终选择了第三种方法。

最终结果为：λ=15 特征维度为 3　测试集损失为 6.90　 收敛时间为 14 秒，迭代次数为 75.

五、总结

通过本次实验，我对最小二乘法在实际问题中的应用有了更加深刻的理解。同时也让自己进一步熟练掌握了 Python 的使用。