姓名：向建宇

学号：1801210687

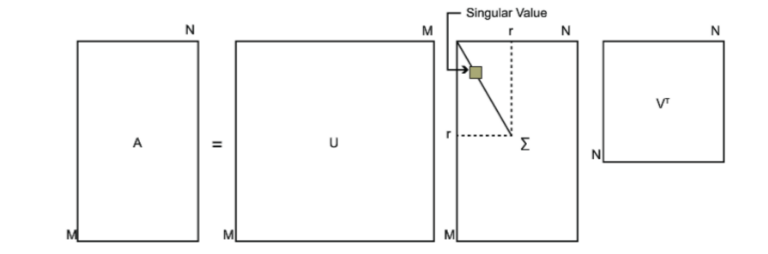
1.用SVD对数据集进行降维并预测：

**算法原理：**

降维是将高维度的数据保留下最重要的一些特征，去除噪声和不重要的特征，从而实现提升数据处理速度的目的。

可以通过数学推理证明，任何矩阵都可以分解成*A*=*U*Σ*VT*，详细推导见：https://www.cnblogs.com/lzllovesyl/p/5243370.html

其可以用如下得形式表达：



即将一个高维得矩阵，通过取90%以上的奇异值，使其变成低维矩阵，大大降低了运算量。

**思路分析：（具体步骤解释请看代码注释）**

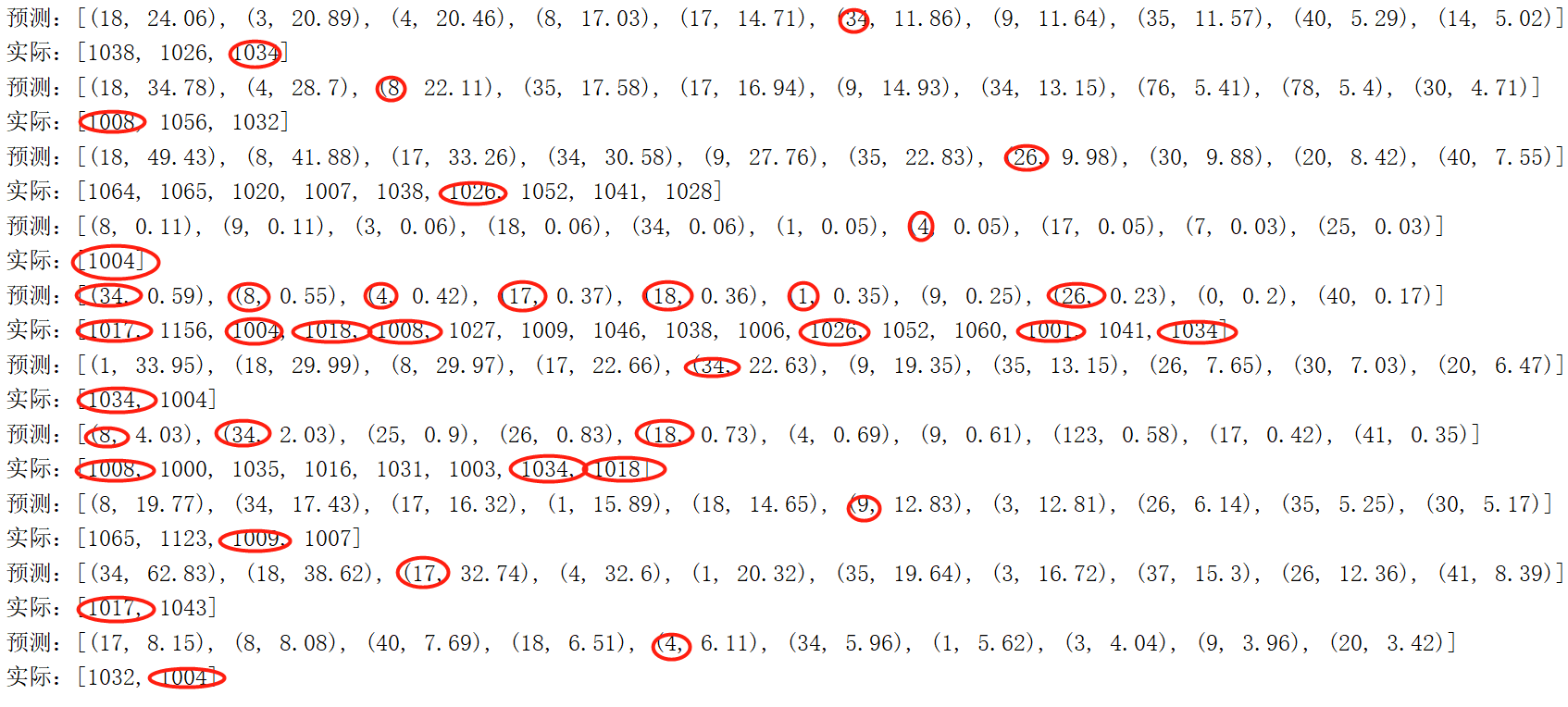
其实最主要的就是根据数学推导，调用库函数得到矩阵分解的U，Sima，V.T的值，然后重构一个低维的矩阵，相似度有两种计算方法，根据用户或者网页，这里用户即行数大概3万行，列数大概300列，比较起来用列数计算相似度要好一点，所以预测评分选用了计算列相似度

其主要步骤如下：

|  |
| --- |
| **第1：加载测试数据集；**  **第2：定义三种计算相似度的方法；**  **第3：通过计算奇异值平方和的百分比来确定将数据降到多少维才合适，返回需要降到的维度；**  **第4：在已经降维的数据中，基于SVD对用户未浏览的网页进行评分预测，返回未打分网页的预测评分值；**  **第5：产生前N个评分值高的网页，返回网页编号以及预测评分值。** |

**结果截图：**

我将维数降低至66维，但由于矩阵实在是太稀疏了，很多行都只有2.3个列，而且把300维降为66维已经只保留了90%的奇异值量，所以不能在下降了，这可能也造成后面预测效果不是太理想，当然还有个原因测试集的数据每个用户的特征太少了，就2.3个，必须把推荐的个数放得足够大才能与预测出来，下面是我得预测值和测试值得比较：



可以看到我将推荐值放大到了10个，每个预测和实际值之间至少有一个交集，而且当实际值增多时，预测值预测正确的个数也变多了（见上图中间），所以这个正确率我不知道该怎么表示了，即若是能预测出来至少一个得话正确率为100%，总体上来讲预测值/实际值不到50%，后面看有时间再进一步优化

完整代码见附录。

附录：

|  |
| --- |
| **import** numpy **as** np *# import tensorflow as tf* **from** scipy.sparse.linalg **import** svds **from** numpy **import** \* **from** numpy **import** linalg **as** la  **""" 三种距离计算模型 """ def** ecludSim(inA,inB):  **return** 1.0/(1.0 + la.norm(inA - inB))  **def** pearsSim(inA,inB):  **if** len(inA) < 3 : **return** 1.0  **return** 0.5+0.5\*corrcoef(inA, inB, rowvar = 0)[0][1]  **def** cosSim(inA,inB):  num = float(inA.T\*inB)  denom = la.norm(inA)\*la.norm(inB)  **return** 0.5+0.5\*(num/denom)  **def** sumSim(inA,inB):  a = array(inA.T)  sum = len(a[0])  **return** sum  **""" Parameters:  fileName - 文件名称 Returns:  re\_mat - 观看记录矩阵  labelMat - 用户名数据集 """ def** loadDataSet(fileName):  fr = open(fileName,**'r'**,encoding = **'utf-8'**)  v\_line,labelMat,dataMat = [],[],[];   **for** line **in** fr.readlines():  lineArr = line.strip().split(**','**) *# 逐行读取，滤除空格等* **if**(lineArr[0] == **"C"**):  labelMat.append(lineArr[2]) *# 添加用户* dataMat.append(v\_line)  v\_line = []  **if**(lineArr[0] == **"V"** ):  v\_line.append(int(lineArr[1]))  **del** dataMat[0]   *#填充30000 \* 296 的数组，形成稀疏矩阵* row = len(labelMat)  re\_mat = np.zeros((row, 296))  i =0  **for** list **in** dataMat:  **for** elem **in** list:  j = elem - 1000  re\_mat[i,j] = 1  i += 1   re\_mat = mat(re\_mat)  *# print(dataMat)  # # print(labelMat)  # # print(re\_mat)* **return** re\_mat, labelMat  **def** loadTestSet(fileName):  fr = open(fileName, **'r'**, encoding=**'utf-8'**)  v\_line, labelMat, dataMat = [], [], [];   **for** line **in** fr.readlines():  lineArr = line.strip().split(**','**) *# 逐行读取，滤除空格等* **if** (lineArr[0] == **"C"**):  labelMat.append(lineArr[2]) *# 添加用户* dataMat.append(v\_line)  v\_line = []  **if** (lineArr[0] == **"V"**):  v\_line.append(int(lineArr[1]))  **del** dataMat[0]  **return** dataMat, labelMat **""" 不降维的标准评分估计 Parameters:  dataMat - 数据矩阵  user - 用户编号  simMeas - 相似度计算方法  item - 物品编号 Returns:  物品的得分 """ def** standEst(dataMat, user, simMeas, item):  n = shape(dataMat)[1]  simTotal = 0.0; ratSimTotal = 0.0  **for** j **in** range(n):  userRating = dataMat[user,j]  **if** userRating == 0: **continue** overLap = nonzero(logical\_and(dataMat[:,item].A>0,dataMat[:,j].A>0))[0] *#寻找两个用户都评级的物品* **if** len(overLap) == 0: similarity = 0 *#返回的索引值数组是一个2维tuple数组，该tuple数组中包含一维的array数组。其中，一维array向量的个数与a的维数是一致的。* **else**: similarity = simMeas(dataMat[overLap,item],dataMat[overLap,j])  *# print('the %d and %d similarity is: %f' % (item, j, similarity))* simTotal += similarity  ratSimTotal += similarity \* userRating  **if** simTotal == 0: **return** 0  **else**: **return** ratSimTotal/100  **""" 奇异值分解SVD后的评分估计 Parameters:  dataMat - 数据矩阵  user - 用户编号  simMeas - 相似度计算方法  item - 物品编号 Returns:  物品的得分 """ def** svdEst(dataMat, user, simMeas, item,xformedItems):  n = shape(dataMat)[1]  simTotal = 0.0; ratSimTotal = 0.0  *# U,Sigma,VT = svds(dataMat) #要用scipy的现代函数处理  # Sig4 = mat(eye(4)\*Sigma[:4]) #arrange Sig4 into a diagonal matrix  # xformedItems = dataMat.T \* U[:,:4] \* Sig4.I #create transformed items  # print(xformedItems)* **for** j **in** range(n):  userRating = dataMat[user,j]  **if** userRating == 0 **or** j==item: **continue** similarity = simMeas(xformedItems[item,:].T,xformedItems[j,:].T)  *# print('the %d and %d similarity is: %f' % (item, j, similarity))* simTotal += similarity  ratSimTotal += similarity \* userRating  **if** simTotal == 0: **return** 0  **else**: **return** ratSimTotal  **def** svd\_tran(dataMat):  U, Sigma, VT = svds(dataMat,70) *# 要用scipy的现代函数处理* Sig4 = mat(eye(66) \* Sigma[:66]) *# arrange Sig4 into a diagonal matrix* xformedItems = dataMat.T \* U[:, :66] \* Sig4.I *# create transformed items* **return** xformedItems **""" 奇异值分解SVD后的评分估计 Parameters:  dataMat -  user -  N=3 -  simMeas=cosSim -  estMethod=standEst - Returns:  物品的得分最高的前N个物品名 """ def** recommend(dataMat, user, N=3, simMeas=cosSim, estMethod=standEst):  unratedItems = nonzero(dataMat[user,:].A==0)[1]*#寻找未评级的物品* **if** len(unratedItems) == 0: **return 'you rated everything'** itemScores = []  xformedItems = svd\_tran(dataMat)   **for** item **in** unratedItems:  *# estimatedScore = estMethod(dataMat, user, simMeas, item,xformedItems)* estimatedScore = estMethod(dataMat, user, simMeas, item)  itemScores.append((item, estimatedScore))  **return** sorted(itemScores, key=**lambda** jj: jj[1], reverse=**True**)[:N] *#寻找前N个未评级物品* **if** \_\_name\_\_ == **'\_\_main\_\_'**:  dataMat, labelMat =loadDataSet(**'anonymous-msweb.data'**)  dataMat\_test, labelMat\_test = loadTestSet(**'anonymous-msweb.test'**)user\_list = [0,1,2,3,4,5,6,7,8,9]i = 0  **for** uesr **in** user\_list:  result = recommend(dataMat, uesr, N=10, simMeas=sumSim, estMethod=standEst)  print(**"预测："**+str(result))  print(**"实际："** + str(dataMat\_test[i]))  i += 1 |