中山大學
SUN YAT-SEN UNIVERSITY

基于奇异值分解方法（SVD）的推荐算法

姓名：钟展辉
学号：15352446

一、算法原理：

1、特征值分解
首先从比较简单的特征值分解说起，对于 $Av = \lambda v$，λ 是矩阵 A 对应于特征向量 v 特征值。特征向量在矩阵 A 的作用下作伸缩运动，伸缩的幅度由特征值决定。λ>1，特征向量伸长；0<λ<1，特征向量缩短；λ<0，特征向量反向。

特征值分解将矩阵 A 分解为：

$A = QEQ^{-1}$，其中 Q 由特征向量组成，E 是由对应特征值组成的对角矩阵。

则 A 可以由矩阵 Q 和矩阵 E 表示。

特征值和特征向量在一定程度上反映了矩阵的总体特征。
而特征向量表示矩阵的一个特征，特征值表示的是这个特征的重要性。因此可以把这些特征值和特征向量按特征值从小大到排序，当矩阵特别大时，可以只取前面的特征值和特征向量，就能较好的表示矩阵特性了，因此 SVD 也能用以降维。

但特征值分解只使用于方阵，而现实应用中很难满足这个条件，因此为了获取这些普通矩阵的重要特征，就用到了奇异值分解。

2、奇异值分解
假设 A 是一个 m*n，即 m 行 n 列的矩阵，则
奇异值分解的形式为：

$$A = UEV^T$$

U 是一个 m*m 的方阵，其列向量两两正交，向量称为左奇异向量；
E 是一个 m*n 的矩阵，除对角线元素外其他都为 0，对角线上元素称为奇异值；
V 是一个 n*n 的方阵，其列向量两两正交，向量称为右奇异向量。

（1）获取V^T：

A^TA是一个方阵，用特征值分解的方法求这个方阵的特征值和特征向量：

$(A^TA)v_i = \lambda_i v_i$

特征向量v_i就是右奇异向量，用这 n 个右奇异向量组成V，再取转置即得到方阵V^T
（一般用施密特正交化等方法使 V 中向量正交）

（2）获取 E：

特征值开方即得到奇异值$\sigma_i = \sqrt{\lambda_i}$
以奇异值作为对角线元素构成 E（非方阵也有对角线）

（3）获取 U：

使用公式$u_i = \frac{1}{\sigma_i} A v_i$ 得到u_i就是左奇异向量，将这 m 个左奇异向量组成矩阵得到 U

（也可以用$(AA^T)u_i = \lambda_i u_i$来求左奇异向量）

稍微验证一下：

由$A = UEV^T$　　$A^T = VEU^T$ 可得：

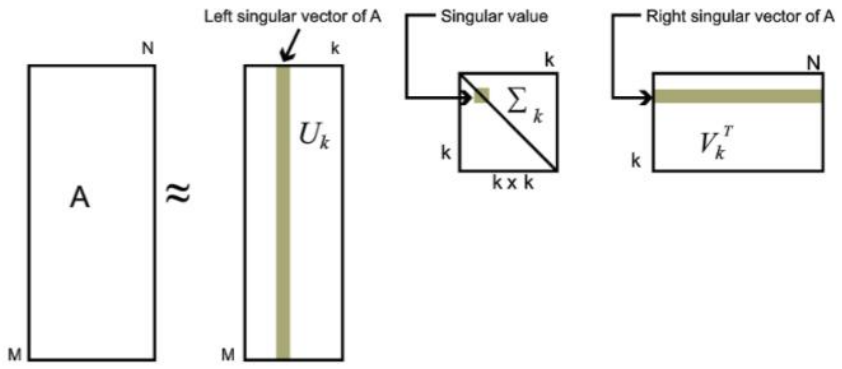
$A^TA = VEU^TUEV^T \xrightarrow{U^TU=I} VE^2V^T$

因此AA^T的特征向量作为 V 的特征向量，而AA^T的特征值开方后作为奇异值

使奇异值σ在矩阵 E 中从大到小排列后，会发现对于一些高维度矩阵，一般情况下奇异值会降低的特别快，奇异值的“贫富差距”十分悬殊，比如前面 10%的奇异值之和可能就已经占据全部奇异值之和的 90%以上了。
也就是说，我们可以用前 k 个奇异值来近似描述矩阵，奇异值分解写成：

$$A_{m*n} \approx U_{m*k} E_{k*k} V^T_{k*n}$$

即只需使用以下图中灰色部分就足够还原出矩阵 A 了（其中 E 中的灰色方块中的奇异值是最大的那部分奇异值集合，U、V 灰色块时对应的特征向量）



SVD 可以理解为：**将一个比较复杂的矩阵用更小更简单的 3 个子矩阵的相乘来表示，这 3 个小矩阵描述了大矩阵重要的特性。**

由于这个重要的性质，SVD 可以用于推荐算法、降维、压缩数据、去噪（使小奇异值为 0）等方面。

3、SVD 应用于推荐系统
将用户和喜好对应的矩阵做特征分解，进而得到隐含的用户需求来做推荐。

矩阵中行代表用户 user，列代表物品 item，其中的值代表用户对物品的打分。基于 SVD 的优势在于：用户的评分数据是稀疏矩阵，可以用 SVD 将原始数据映射到低维空间中，然后计算物品 item 之间的相似度，可以节省计算资源。

用户\物品	物品1	物品2	物品3	物品4	物品5	物品6	物品7
用户1	3		5			1	
用户2		2					4
用户3				4			
用户4			2				1
用户5	1				4		

我们希望预测目标用户对其他未评分物品的评分，进而将评分高的物品推荐给目标用户

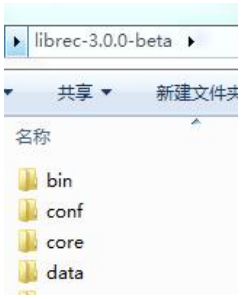
比较简单的两种实现方式：
（1）预先对矩阵中未评分项进行填充，比如填充全局平均值，然后用 SVD 对评分矩阵进行分解，得到三个矩阵，取最重要的 k 个奇异值降维，然后三个矩阵再相乘重构出评分矩阵，此时评分矩阵是每个都有值的，取评分值最高且用户未使用过的几个物品推荐给用户。

（2） 降维之后得到 U（m x k）和 I（k x n）矩阵，其中 $I = EV^T$；

SVD 降维的过程不仅是降低了维度，还进行了自动的特征提取。
①矩阵 U（m x k）可以用来做 user-based（基于 m 个 user 的 k 个 feature 来进行相似度计算）
②矩阵 I（k x n）可以用来做 item-based 推荐（基于 n 个 item 的 k 个 feature 来进行相似度计算）

二、实验过程

1、从网站 https://www.librec.net/上下载源代码，截取部分如下：



2、调出命令行进入 bin 目录下，使用语句
librec rec -exec -D rec.recommender.class=svdpp 使用 SVD++算法

BiasedMFRecommender → MatrixFactorizationRecommender	cf.rating	svdpp	SVDPlusPlusRecommender

其中默认运行设置在 conf 文件夹中的 librec.properties 文件中：可见输入是 data/filmtrust/rating 目录下的文件，输出目录是 result：

librec.properties

```
# set data directory
dfs.data.dir=../data
# set result directory
# recommender result will output in this folder
dfs.result.dir=../result

# convertor
# load data and splitting data
# into two (or three) set
# setting dataset name
data.input.path=filmtrust/rating
# setting dataset format(UIR, UIRT)
data.column.format=UIR
```

运行结果截图：

```
18/07/25 17:39:09 INFO TextDataConvertor: Dataset: ../data/filmtrust/rating
18/07/25 17:39:09 INFO TextDataConvertor: All dataset files [..\data\filmtrust\rating\ratings_0.txt, ..\data\filmtrust\rating\rating
18/07/25 17:39:09 INFO TextDataConvertor: All dataset files size 411942
18/07/25 17:39:09 INFO TextDataConvertor: Now loading dataset file ratings_0
18/07/25 17:39:09 INFO TextDataConvertor: Now loading dataset file ratings_1
18/07/25 17:39:09 INFO TextDataConvertor: Now loading dataset file ratings_2
18/07/25 17:39:09 INFO TextDataConvertor: Now loading dataset file ratings_3
18/07/25 17:39:09 INFO TextDataModel: Transform data to Convertor successfully!
18/07/25 17:39:10 INFO TextDataModel: Split data to train Set and test Set successfully!
18/07/25 17:39:10 INFO TextDataModel: Data size of training is 28408
18/07/25 17:39:10 INFO TextDataModel: Data size of testing is 7086
18/07/25 17:39:10 INFO SVDPlusPlusRecommender: Job Setup completed.
18/07/25 17:39:11 INFO SVDPlusPlusRecommender: SVDPlusPlusRecommender iter 1: loss = 11201.423045450894, delta_loss = -11201.423
18/07/25 17:39:11 INFO SVDPlusPlusRecommender: SVDPlusPlusRecommender iter 2: loss = 9907.901131301518, delta_loss = 1293.522
18/07/25 17:39:11 INFO SVDPlusPlusRecommender: SVDPlusPlusRecommender iter 3: loss = 9263.318464005864, delta_loss = 644.58264
18/07/25 17:39:11 INFO SVDPlusPlusRecommender: SVDPlusPlusRecommender iter 4: loss = 8881.129931453039, delta_loss = 382.18854
18/07/25 17:39:11 INFO SVDPlusPlusRecommender: SVDPlusPlusRecommender iter 5: loss = 8622.452595038158, delta_loss = 258.67734
18/07/25 17:39:11 INFO SVDPlusPlusRecommender: SVDPlusPlusRecommender iter 6: loss = 8421.183975806576, delta_loss = 201.26862
18/07/25 17:39:12 INFO SVDPlusPlusRecommender: SVDPlusPlusRecommender iter 7: loss = 8243.650395066607, delta_loss = 177.53358
18/07/25 17:39:12 INFO SVDPlusPlusRecommender: SVDPlusPlusRecommender iter 8: loss = 8071.882164975898, delta_loss = 171.76823
18/07/25 17:39:12 INFO SVDPlusPlusRecommender: SVDPlusPlusRecommender iter 9: loss = 7896.353656744996, delta_loss = 175.5285
18/07/25 17:39:12 INFO SVDPlusPlusRecommender: SVDPlusPlusRecommender iter 10: loss = 7712.698072828891, delta_loss = 183.65558
18/07/25 17:39:12 INFO SVDPlusPlusRecommender: SVDPlusPlusRecommender iter 11: loss = 7519.98709976835, delta_loss = 192.71097
18/07/25 17:39:12 INFO SVDPlusPlusRecommender: SVDPlusPlusRecommender iter 12: loss = 7319.798290174444, delta_loss = 200.18881
18/07/25 17:39:12 INFO SVDPlusPlusRecommender: SVDPlusPlusRecommender iter 13: loss = 7115.466701377677, delta_loss = 204.33159
18/07/25 17:39:12 INFO SVDPlusPlusRecommender: Job Train completed.
18/07/25 17:39:12 INFO SVDPlusPlusRecommender: Job End.
18/07/25 17:39:12 INFO RecommenderJob: Evaluator value:MPE is 0.9801016088060965
18/07/25 17:39:12 INFO RecommenderJob: Evaluator value:RMSE is 0.8012644628336554
18/07/25 17:39:12 INFO RecommenderJob: Evaluator value:MSE is 0.6420247394001063
18/07/25 17:39:12 INFO RecommenderJob: Evaluator value:MAE is 0.6159432678611797
18/07/25 17:39:12 INFO RecommenderJob: Result path is ../result/filmtrust/rating-svdpp-output/svdpp
```

可见输入文件被分割出 28408 个训练集样本和 7086 个测试集样本，最终的均方根误差 RMSE=0.8013

三、实验总结

1、SVD 优缺点

（1）SVD 优点在于原理简单，只涉及到基本的线性代数知识，实现也很比较容易，而预测效果较好。

（2）SVD 的缺点是相比于基于概率的逻辑回归之类的推荐算法，SVD 分解出的矩阵解释性不强；此外矩阵分解前要对缺失值作填充，不同填充值效果可能相差很远，而难以选择填充值；然后当评分矩阵十分庞大时，SVD 算法将出现耗时长的问题。

2、SVD 适用场景

根据 SVD 的特性，知道 SVD 使用于中小型的推荐系统，而对于规模庞大的应用场景，则 SVD 不如深度学习模型。

3、改进方向

（1）传统的 SVD 算法使用矩阵运算可能会耗时过长，我们可以使用梯度下降的方式进行矩阵分解：假设评分矩阵能被分解为 U（m x k）和 I（k x n）矩阵，先初始化 U 和 I 的值，U 和 I 相乘得到评分矩阵（m x n），将之与真实的评分进行比较（甚至可以不考虑填充的问题），计算误差 loss，根据 loss 的梯度反复迭代更新 U 和 I。

（2）SVD++，

不仅仅考虑用户评分，还增添了更多的考量因素，比如用户偏见，有的用户给分总是偏高或偏低、有的物品得到的评分总是偏高或偏低等。此外还有隐式反馈，比如用户给某个物品评分了，这个评分是显式反馈，而用户使用过这个物品则是一种隐式反馈，也可以纳入考量。

主要参考资料：

1、《机器学习实战》
2、http://www.ams.org/publicoutreach/feature-column/fcarc-svd
3、https://www.cnblogs.com/pinard/p/6251584.html
4、https://blog.csdn.net/scutdzj/article/details/54018738
5、https://www.cnblogs.com/pinard/p/6351319.html
6、https://blog.csdn.net/github_36326955/article/details/55051489?locationNum=7&fps=1