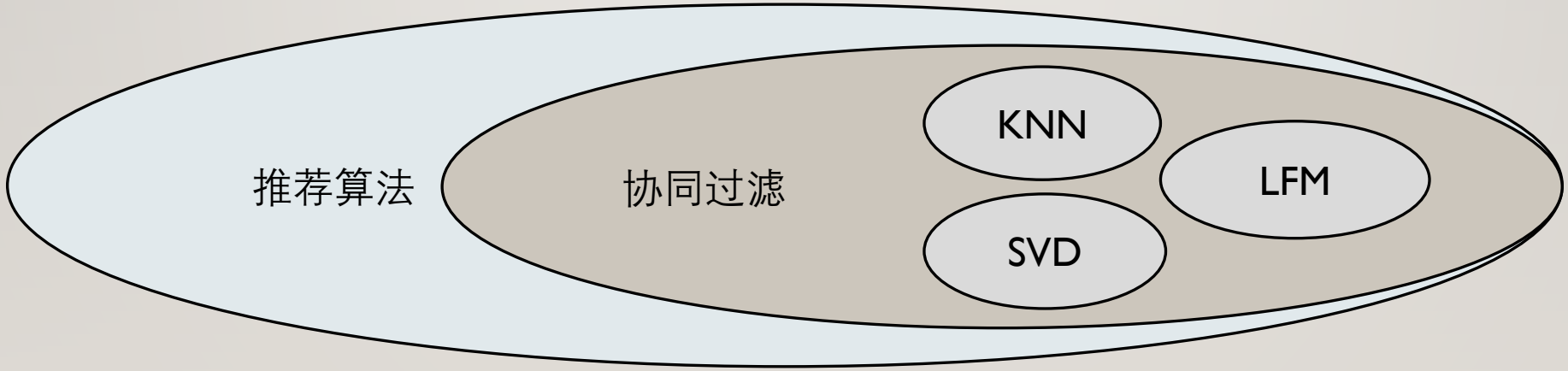
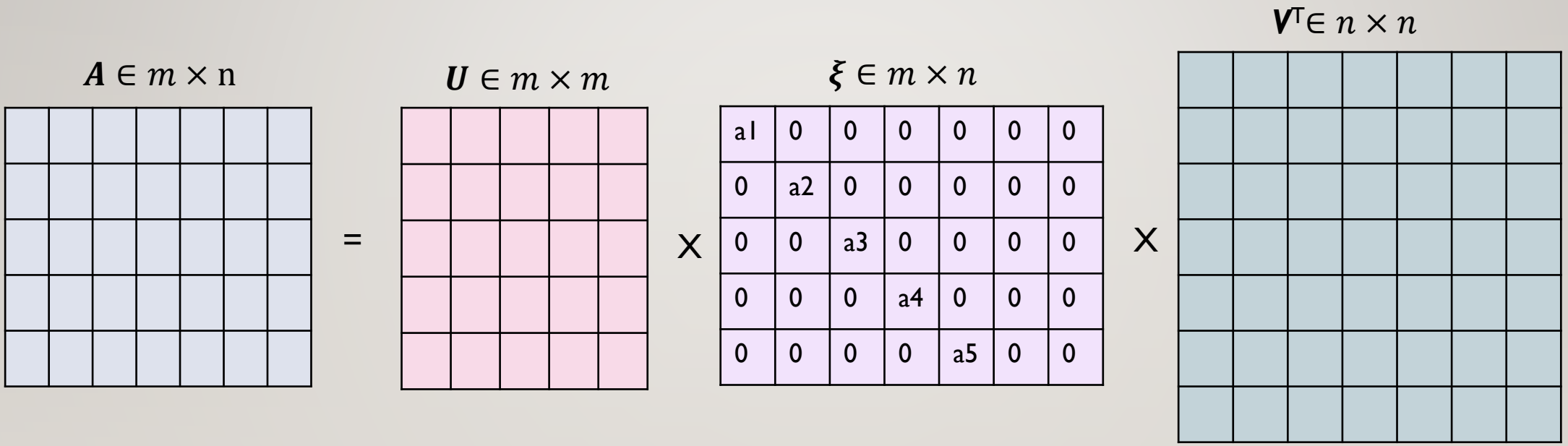
SVD与LFM推荐系统

- 1. SVD矩阵分解
- 2. LFM(latent factor model), 又被称 为ALS (alternating least squares)
- 3. 推荐算法, 协同过滤, SVD, LFM, KNN的包含关系:

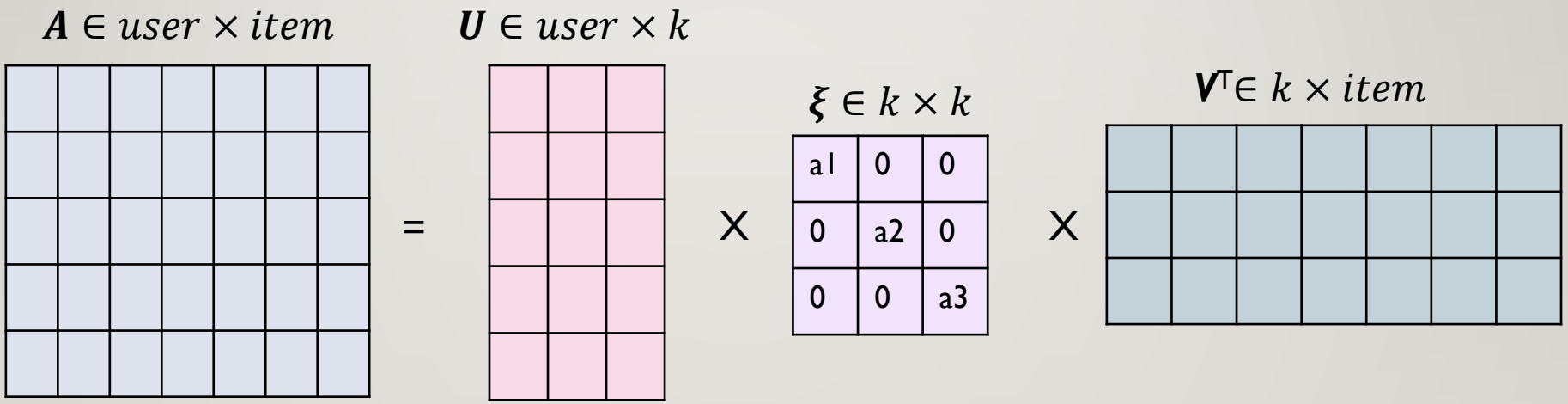


SVD推荐系统



- SVD 矩阵分解就是将一个m*n的矩阵分解成如图所示的三个矩阵,其中ξ 是仅在主对角线有值其余元素为0的矩阵, 那些值被称为奇异值。

SVD推荐系统



- 奇异值的大小代表对应U,V矩阵中对应位置元素的重要性,且奇异值会从大到小排列。我们可以取前k个奇异值然后将原来的矩阵降维成如图所示。

- *SVD分解常用作矩阵降维，或者图像压缩。*

SVD推荐系统

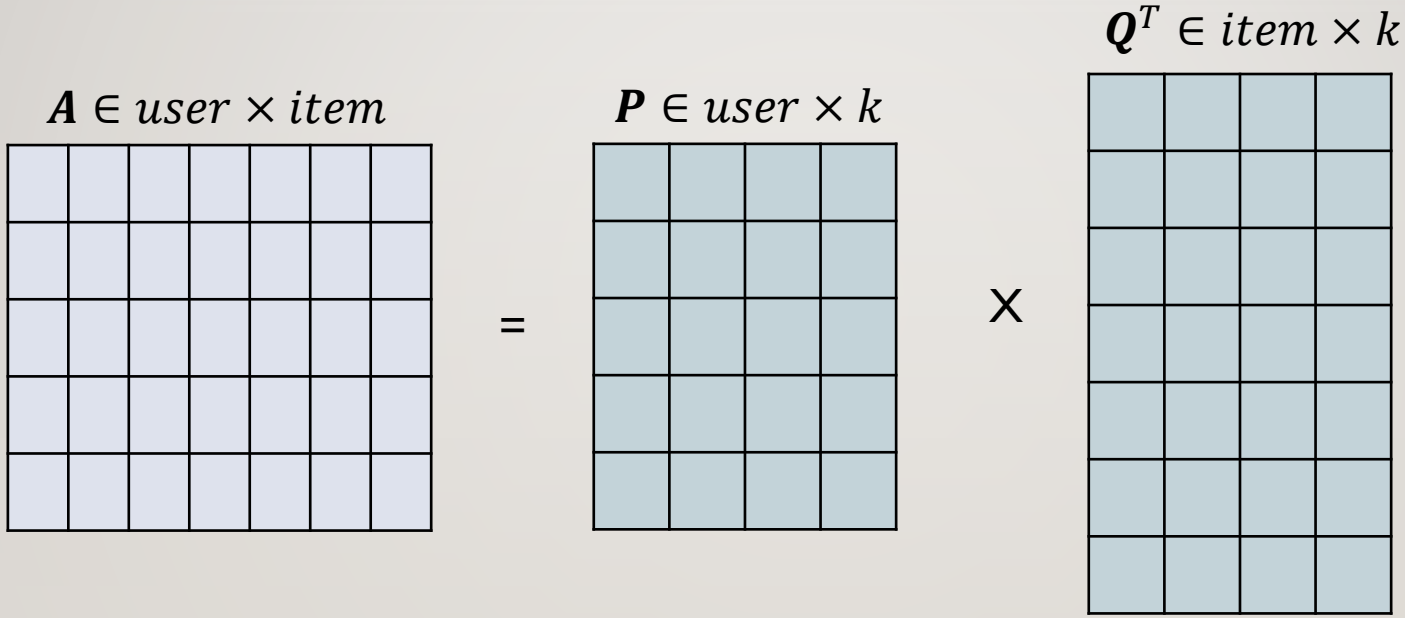
- SVD推荐系统思路:

1. 得到user与item的共现矩阵。
2. 采用SVD降维得到user*k, item*k,及k*k的奇异值对角矩阵。
3. 如要进行某个user对某个item评分, 则 评分R = $u\xi v^T$ 。
   其中u为此user在user矩阵中对应位置的向量, v为此item在item矩阵中对应位置的向量。

- SVD推荐的缺点:

1. 需要内存中载入共现矩阵。
2. 不定义位置需要想个策略去初始化。
3. 由于以上两个缺点，实际工作中几乎不会有纯SVD的推荐系统。

LFM推荐系统



- LFM推荐系统是指将user*item数量的共现矩阵,分解成user*k及item*k的两个矩阵。这两个矩阵中对应的向量可被称为该user或item的隐因子。

LFM推荐系统

- LFM 全称 latent factor model, 隐因子模型。
- ALS 全称 alternating least squares, 交替最小二乘。
- 训练方式:

1. 随机初始化 P Q 两个矩阵。
2. 将训练数据中u 和 i的评分与对应的P_u与Q_i^T 点乘得到的值建立平方差损失函数。
3. 用梯度下降优化损失函数。
4. 最后输出训练好的PQ 矩阵。

LFM推荐系统

- 全局目标函数: $A = P \cdot Q^T$
- 预测单个u与单个i的评分: $r_{ui}^{hat} = P_u \cdot Q_i^T$
- 损失函数: $error_0 = \sum_{(u,i)\in A}(r_{ui} - r_{ui}^{hat})\text{^}2$
- 加上正则的损失函数: $error = \sum_{(u,i)\in A}(r_{ui} - P_u \cdot Q_i^T)^2 + \lambda\left(||P_u||^2 + ||Q_i||^2\right)$

  λ为正则项系数
- P_u的偏导 $= -2 * (r_{ui} - P_u \cdot Q_i^T) * Q_i + 2\,\lambda P_u$
- Q_i的偏导 $= -2 * (r_{ui} - P_u \cdot Q_i^T) * P_u + 2\,\lambda Q_i$

LFM推荐系统

- 协同过滤普遍有个问题是user打分偏差与item流行度长尾问题。
- 用户打分偏差是指如果某个user给任何item都是打高分，则他的高分可能也就没那么重要。同理，某个user如果给任何item都是打低分，则他的低分也就没那么重要。
- Item流行度长尾问题是指热门item会更多的与用户发生关系，则相较热门item会得到更多的表现分。则更容易被推荐出来，则会更热门。同理，冷门的会更冷门。如不加控制，则会呈现流行度的长尾分布。
- 我们可以给每个user及item增加偏置项系数，从而解决上述问题。

LFM推荐系统

- 包含偏置项预测单个u与单个i的评分公式: $r_{ui}^{hat} = P_u \cdot Q_i^T + \mathrm{b_u} + \mathrm{b_i}$
- 包含偏置项与正则项的损失函数:

$$error = \sum_{(u,i)\in A} (r_{ui} - P_u \cdot Q_i^T - \mathrm{b_u} - \mathrm{b_i})^2 + \lambda \left(||P_u||^2 + ||Q_i||^2 + \mathrm{b_u^2} + \mathrm{b_i^2} \right)$$

- P_u的偏导 $= -2 * (r_{ui} - P_u \cdot Q_i^T - \mathrm{b_u} - \mathrm{b_i}) * Q_i + 2\lambda P_u$
- Q_i的偏导 $= -2 * (r_{ui} - P_u \cdot Q_i^T - \mathrm{b_u} - \mathrm{b_i}) * P_u + 2\lambda Q_i$
- $\mathrm{b_u}$的偏导 $= -2 * (r_{ui} - P_u \cdot Q_i^T - \mathrm{b_u} - \mathrm{b_i}) + 2\lambda \mathrm{b_u}$
- $\mathrm{b_i}$的偏导 $= -2 * (r_{ui} - P_u \cdot Q_i^T - \mathrm{b_u} - \mathrm{b_i}) + 2\lambda \mathrm{b_i}$

评测指标

- $MSE = \frac{1}{|A|}\sum_{(u,i)\in A}(r_{ui} - r_{ui}^{hat})\text{^}2$

Mean Squared Error 均方误差

- $RMSE = \sqrt{MSE}$

Root Mean Squared Error 均方根误差

- $MAE = \frac{1}{|A|}\sum_{(u,i)\in A}|r_{ui} - r_{ui}^{hat}|$

Mean Absolute Error 平均绝对误差

推荐系统PYTHON API推荐

一. pyspark: http://spark.apache.org/docs/latest/api/python/pyspark.ml.html#pyspark.ml.recommendation.ALS

Spark 会有两个机器学习库ml和mlib, 推荐使用ml, 理由:

(1) mlib会全面废弃。

(2) ml基于Dataframe 类似于pandas的Dataframe 操作方便, 耦合度低; mlib基于RDD 操作麻烦。

二. surprise:

a) 欢迎界面: http://surpriselib.com/

b) 文档界面: https://surprise.readthedocs.io

更轻量级，更专业，专门用来做推荐系统的库。

LFM推荐系统的优缺点

优点:

1. 泛化能力强。
2. 空间复杂度低。不需要以稀疏矩阵的形式加载数据，且最后模型中只需储存隐向量，空间复杂度由 user_count * item_count , 降低到 (user_count + item_count) * k 。
3. 更好的扩展性和灵活性。隐向量与深度学习中的embedding思想不谋而合，也便于和其它特征进行组合和拼接。
4. 协同过滤普遍的优点是不需要收集用户及物品本身的特征属性，仅凭用户与物品发生关系产生的数据就可以做推荐系统。

LFM推荐系统的优缺点

缺点:

1. 前一页第四个优点也是它的缺点。协同过滤不需要也不方便加入用户和物品本身的特征及上下文相关的特征。
2. 无历史行为数据时无法进行推荐,也就是说cover不了用户冷启动及物品冷启动。