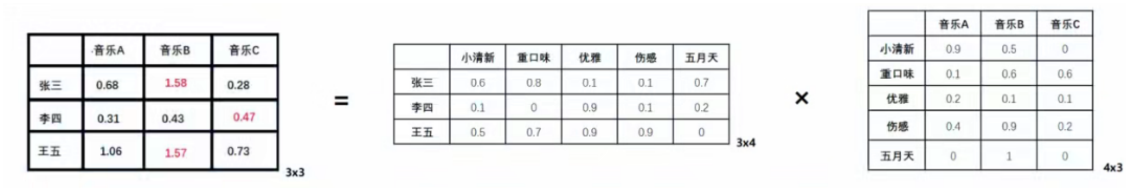
MF矩阵分解 -- SVD、LFM、RSVD、SVD++（Matrix Factorization）

- **针对问题：**

  协同过滤处理稀疏矩阵的能力较弱

  协同过滤中，相似度矩阵维护难度大

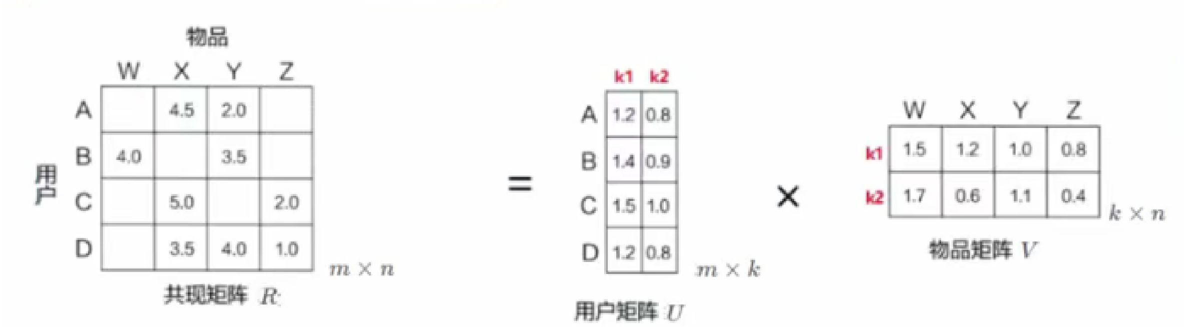
- **解决思路：**



	音乐A	音乐B	音乐C
张三	0.68	1.58	0.28
李四	0.31	0.43	0.47
王五	1.06	1.57	0.73

	小清新	重口味	优雅	伤感	五月天
张三	0.6	0.8	0.1	0.1	0.7
李四	0.1	0	0.9	0.1	0.2
王五	0.5	0.7	0.9	0.9	0

	音乐A	音乐B	音乐C
小清新	0.9	0.5	0
重口味	0.1	0.6	0.6
优雅	0.2	0.1	0.1
伤感	0.4	0.9	0.2
五月天	0	1	0



	W	X	Y	Z
A		4.5	2.0	
B	4.0		3.5	
C		5.0		2.0
D		3.5	4.0	1.0

	k1	k2
A	1.2	0.8
B	1.4	0.9
C	1.5	1.0
D	1.2	0.8

	W	X	Y	Z
k1	1.5	1.2	1.0	0.8
k2	1.7	0.6	1.1	0.4

-
  - 隐含特征是不可解释的，需要模型自己学习
  - k的大小决定隐向量表达能力强弱，k越大表达能力越强，用户兴趣和物品分类具体
  - 通过用户矩阵和物品矩阵预测评分计算公式：

    $$Preference(n,i) = r_{ui} = \sum_{f=1}^{F} p_{u,k} q_{k,i}$$ （对应向量内积）

- **MF方式**
  - **特征值分解**
    - 特征值，特征向量： $Av = \lambda v$

      v是特征向量，λ是特征向量
    - 特征值分解： $A = Q \sum Q^{-1}$

      Q代表矩阵A的特征向量构成的矩阵

      $\sum$是对角阵，对角线的元素是特征值
  - **奇异值分解（SVD）：**
    - 定义： $A = U \sum V^T$

      其中　A是实矩阵，$UU^T = I, VV^T = I$

      $\sum$是对角矩阵，对角线元素非负且降序排列

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix}_{5\times4} = \begin{bmatrix} 0 & 0 & \sqrt{0.2} & 0 & \sqrt{0.8} \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \sqrt{0.8} & 0 & -\sqrt{0.2} \end{bmatrix}_{5\times5} \times \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & \sqrt{5} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}_{5\times4} \times \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}_{4\times4}$$

A $\quad$ U $\quad$ Σ $\quad$ V^{T}

$UU^{\mathrm{T}} = I_5$，

对角线外的元素都是 0，
对角线上的元素非负，
按降序排列。

$VV^{\mathrm{T}} = I_4$

- 计算步骤

  A是一个m*n的实矩阵

  1. 构造n阶实对称矩阵 $W = A^T A$
  2. 计算W的特征值与特征向量

     求解特征方程

     $$(W - \lambda I)x = 0$$

     得到特征值 λ_i，并将特征值由大到小排列

     $$\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_n \geqslant 0$$

     将特征值 λ_i（$i = 1, 2, \cdots, n$）代入特征方程求得对应的特征向量。

  3. 求得n阶正交矩阵V

     将特征向量单位化，得到单位特征向量 $v_1, v_2, \cdots, v_n$，构成 n 阶正交矩阵 V：

     $$V = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix}$$

  4. 求得m*n对角矩阵

     计算 A 的奇异值

     $$\sigma_i = \sqrt{\lambda_i}, \quad i = 1, 2, \cdots, n$$

     构造 $m \times n$ 矩形对角矩阵 Σ，主对角线元素是奇异值，其余元素是零，

     $$\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \cdots, \sigma_n)$$

  5. 求得m阶正交矩阵U（求得上述）

     - 求U1

       对 A 的前 r 个正奇异值，令

       $$u_j = \frac{1}{\sigma_j} A v_j, \quad j = 1, 2, \cdots, r$$

       得到

       $$U_1 = \begin{bmatrix} u_1 & u_2 & \cdots & u_r \end{bmatrix}$$

     - 求U2

       求 A^{T} 的零空间的一组标准正交基 $\{u_{r+1}, u_{r+2}, \cdots, u_m\}$，令

       $$U_2 = \begin{bmatrix} u_{r+1} & u_{r+2} & \cdots & u_m \end{bmatrix}$$

        - 得到U=[U1, U2]
    - 缺点：

      传统SVD分解会要求原始矩阵是稠密的，所以我们需要对缺失进行填补，空间复杂度非常高，基本无法解决大规模稀疏矩阵的矩阵分解问题

- **Basic SVD（LFM，Funk SVD）**
    - 将矩阵分解问题转化为**最优化问题**，通过梯度下降进行优化
    - 预测函数：

$$\text{Preference}(u,i) = r_{ui} = p_u^T q_i = \sum_{f=1}^{F} p_{u,k} q_{k,i}$$

    - 损失函数（误差平方和）：

      式子2便于优化

$$\text{SSE} = \sum_{u,i} e_{ui}^2 = \sum_{u,i} \left(r_{ui} - \sum_{k=1}^{K} p_{u,k} q_{k,i} \right)^2$$

$$\text{SSE} = \frac{1}{2} \sum_{u,i} e_{ui}^2 = \frac{1}{2} \sum_{u,i} \left(r_{ui} - \sum_{k=1}^{K} p_{uk} q_{ki} \right)^2$$

    - 步骤：
        1. 首先先初始化这两个矩阵
        2. 把用户评分矩阵里面已经评过分的那些样本当做训练集的label， 把对应的用户和物品的隐向量当做features， 这样就会得到(features, label)相当于训练集
        3. 通过两个隐向量乘积得到预测值pred
        4. 根据label和pred计算损失
        5. 然后反向传播， 通过梯度下降的方式，更新两个隐向量的值
        6. 未评过分的那些样本当做测试集， 通过两个隐向量就可以得到测试集的label值
        7. 这样就填充完了矩阵， 下一步就可以进行推荐了
- **RSVD**

  在Basic SVD基础上，加入正则化参数（惩罚项）

  预测函数：

$$\text{Preference}(u,i) = r_{ui} = p_u^T q_i = \sum_{f=1}^{F} p_{u,k} q_{k,i}$$

  目标函数：

$$SSE = \frac{1}{2}\sum_{u,i} e_{ui}^2 + \frac{1}{2}\lambda\sum_{u} |p_u|^2 + \frac{1}{2}\lambda\sum_{i} |q_i|^2$$

$$= \frac{1}{2}\sum_{u,i} e_{ui}^2 + \frac{1}{2}\lambda\sum_{u}\sum_{k=0}^{K} p_{u,k}^2 + \frac{1}{2}\lambda\sum_{i}\sum_{k=0}^{K} q_{k,i}^2$$

-
  - **改进（LFM）：**

    Netflix提出另外一种LFM，在原有基础上加偏置项，消除用户和物品打分的偏差

    原因：不同用户打分体系不同，不同物品衡量标准有区别，导致评分偏差

    $$\hat{r}_{ui} = \underbrace{\mu}_{\text{评分的全局平均数}} + \underbrace{b_u}_{\text{用户评分偏差}} + \underbrace{b_i}_{\text{物品得分偏差}} + p_u^T \cdot q_i$$

    目标函数:

$$\mathrm{SSE} = \tfrac{1}{2}\sum_{u,i} e_{ui}^2 + \tfrac{1}{2}\lambda\sum_{u} |\boldsymbol{p}_u|^2 + \tfrac{1}{2}\lambda\sum_{i} |\boldsymbol{q}_i|^2 + \tfrac{1}{2}\lambda\sum_{u} \boldsymbol{b}_u^2 + \tfrac{1}{2}\lambda\sum_{u} \boldsymbol{b}_i^2$$
$$= \tfrac{1}{2}\sum_{u,i}\left(\boldsymbol{r}_{ui} - \boldsymbol{\mu} - \boldsymbol{b}_u - \boldsymbol{b}_i - \sum_{k=1}^{K} \boldsymbol{p}_{uk}\boldsymbol{q}_{ki}\right)^2 + \tfrac{1}{2}\lambda\sum_{u} |\boldsymbol{p}_u|^2 + \tfrac{1}{2}\lambda\sum_{i} |\boldsymbol{q}_i|^2 + \tfrac{1}{2}\lambda\sum_{u} \boldsymbol{b}_u^2 + \tfrac{1}{2}\lambda\sum_{u} \boldsymbol{b}_i^2$$

  - **SVD++**

    改进方向：用户历史记录会对新评分产生影响（即物品间存在某些联系），交给模型学习

    首先先把ItemCF的预测算法改成一个可以学习的模型，就行LFM那样，怎么改？ItemCF的预测算法公式如下：

    $$\hat{r}_{ui} = \frac{1}{\sqrt{|N(u)|}} \sum_{j\in N(u)} w_{ij}$$

    还记得ItemCF吗？这个式子是预测用户u对于物品i的打分，$N(u)$表示用户u打过分的历史物品，w_{ij}表示物品ij的相似度，当然这里的这个相似度不再是ItemCF那样，通过向量计算的，而是想向LFM那样，让模型自己学出这个参数来，那么相应的就可以通过优化的思想嘛：

    $$SSE = \sum_{(u,i)\in \mathrm{Train}} \left(r_{ui} - \sum_{j\in N(u)} w_{ij} r_{uj}\right)^2 + \lambda w_{ij}^2$$

    但是呢，这么模型有个问题，就是w比较稠密，存储需要很大的空间，因为如果有n个物品，那么模型的参数就是n^2，参数一多，就容易造成过拟合。所以Koren提出应该对w矩阵进行分解，将参数降到了$2*n*F$：

    $$\hat{r}_{ui} = \frac{1}{\sqrt{|N(u)|}} \sum_{j\in N(u)} x_i^T y_j = \frac{1}{\sqrt{|N(u)|}} x_i^T \sum_{j\in N(u)} y_j$$

    相当于用$x_i^T y_j$代替了w_{ij}，这里的x_i, y_j是两个F维的向量。**有没有发现在这里，就出现了点FM的改进身影了。**这里其实就是又对物品i和某个用户u买过的历史物品又学习一波隐向量，这次是F维，为了衡量出物品i和历史物品j之间的相似性来。这时候，参数的数量降了下来，并同时也考虑进来了用户的历史物品记录。所以这个和之前的LFM相加就得到了：

    $$\hat{r}_{ui} = \mu + b_u + b_i + p_u^T \cdot q_i + \frac{1}{\sqrt{|N(u)|}} x_i^T \sum_{j\in N(u)} y_j$$

前面的是我们之前分析的LFM模型， 而后面的这个是考虑进了用户购买的历史物品。 但是这样感觉参数太多了， 所以Koren提出令$x = q$, 因为既然同是商品i， 就没有必要学习两个隐向量了嘛， 所以得到了该模型的最终预测方式：

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + \frac{1}{\sqrt{|N(u)|}} \sum_{j \in N(u)} y_j \right)$$

这一个就是SVD++模型了。 有了预测函数， 然后也知道真实值， 就可以由损失函数对各个参数求偏导， 和上面的一样了， 这里直接给出导数了， 不推了：

$$e_{ui} = r_{ui} - \hat{r}_{ui}$$
$$b_u \leftarrow b_u + \gamma \cdot (e_{ui} - \lambda \cdot b_u)$$
$$b_i \leftarrow b_i + \gamma \cdot (e_{ui} - \lambda \cdot b_i)$$
$$p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u)$$
$$q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot (p_u + \frac{1}{\sqrt{\|R_u\|}} \textstyle\sum_{j \in R_u} y_j) - \lambda \cdot q_i)$$
$$y_j \leftarrow y_j + \gamma (e_{ui} \cdot \frac{1}{\sqrt{\|R_u\|}} \cdot q_i - \lambda \cdot q_i)$$