推荐系统系列之二：矩阵分解

作者：周秀泽

邮箱：zhouxiuze@foxmail.com

程序地址：https://github.com/XiuzeZhou/Recommender-Systems

本文的 PDF 下载地址：https://github.com/XiuzeZhou/Recommender-Systems/tree/master/pdf

1. 理论基础

说明介绍：

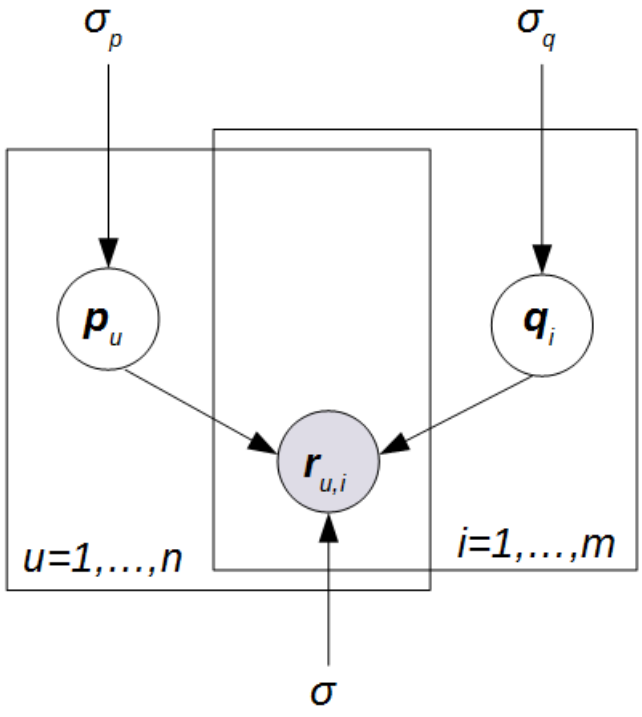
从数学概率的角度，证明了 MF 的由来。这样使得 概率矩阵分解（PMF） 和其他模型的“搭配”有了理论的依据。

来源出处：

- Salakhutdinov et al. Probabilistic matrix factorization. NIPS(2008): 1257-1264.

定义和描述

假设现在有 n 个用户， m 个商品，形成一个 $n \times m$ 维的评分矩阵 $\mathbf{R}$，其中的元素 $r_{u,i}$ 表示用户 u 对商品 i 的评分。假设潜在特征个数为 k，那么 $n \times k$ 维的 $\mathbf{p}$ 表示用户的潜在特征矩阵，其中 $\mathbf{p}_u$ 表示用户 u 的潜在特征向量； $m \times k$ 维的矩阵 $\mathbf{q}$ 表示商品的潜在特征矩阵，其中 $\mathbf{q}_i$ 商品 i 的潜在特征向量。概率模型图如下图所示：



主要推导：

假设关于已知评分数据的条件分布满足高斯分布：

$$p\left(\mathbf{R}|\mathbf{p},\mathbf{q},\sigma^2\right)=\prod_{u=1}^{n}\prod_{i=1}^{m}\left[N\left(r_{u,i}|\mathbf{p}_u\mathbf{q}_i^T,\sigma^2\right)\right]^{I_{ij}},$$

其中，$I_{u,i}$ 表示指示函数，当用户 u 与商品 i 有互动时，$I_{u,i}=1$，否则为0。

再假设用户潜在特征向量和商品潜在特征向量都服从均值为 0 的高斯先验分布，即：

$$p\left(\mathbf{p}|\sigma_p^2\right)=\prod_{u=1}^{n}N\left(\mathbf{p}_u|0,\sigma_p^2\mathbf{I}\right),p\left(\mathbf{q}|\sigma_q^2\right)=\prod_{i=1}^{m}N\left(\mathbf{q}_i|0,\sigma_q^2\mathbf{I}\right).$$

注意这个公式中的 $\mathbf{I}$ 不是指示函数，表示一个对角阵。

然后，计算 $\mathbf{p}$ 和 $\mathbf{q}$ 的后验概率：

$$p\left(\mathbf{p},\mathbf{q}|R,\sigma^2,\sigma_q^2,\sigma_p^2\right)=\frac{p\left(\mathbf{p},\mathbf{q},R,\sigma^2,\sigma_q^2,\sigma_p^2\right)}{p\left(R,\sigma^2,\sigma_q^2,\sigma_p^2\right)}=\frac{p\left(R|\mathbf{p},\mathbf{q},\sigma^2\right)\times p\left(\mathbf{p},\mathbf{q}|\sigma_q^2,\sigma_p^2\right)}{p\left(R,\sigma^2,\sigma_q^2,\sigma_p^2\right)}$$

$$\sim p\left(R|\mathbf{p},\mathbf{q},\sigma^2\right)\times p\left(\mathbf{p},\mathbf{q}|\sigma_q^2,\sigma_p^2\right)$$

$$=p\left(R|\mathbf{p},\mathbf{q},\sigma^2\right)\times p\left(\mathbf{p}|\sigma_p^2\right)\times p\left(\mathbf{q}|\sigma_q^2\right)$$

$$=\prod_{u=1}^{n}\prod_{i=1}^{m}\left[N\left(r_{u,i}|\mathbf{p}_u\mathbf{q}_i^T,\sigma^2\right)\right]^{I_{u,i}}\times\prod_{u=1}^{n}\left[N\left(\mathbf{p}_u|0,\sigma_p^2I\right)\right]\times\prod_{i=1}^{m}\left[N\left(\mathbf{q}_i|0,\sigma_q^2I\right)\right]$$

等式两边取对数 ln 后得到：

$$lnp\left(\mathbf{p},\mathbf{q}|\mathbf{R},\sigma^2,\sigma_p^2,\sigma_q^2\right)=-\frac{1}{2\sigma^2}\sum_{u=1}^{n}\sum_{i=1}^{m}I_{ij}\left(r_{u,i}-\mathbf{p}_u\mathbf{q}_i^T\right)^2-\frac{1}{2\sigma_p^2}\sum_{i=1}^{N}\mathbf{p}_u\mathbf{p}_u^T-\frac{1}{2\sigma_q^2}\sum_{i=1}^{M}\mathbf{q}_i\mathbf{q}_i^T$$

$$-\frac{1}{2}\left(\left(\sum_{i=1}^{n}\sum_{j=1}^{m}I_{u,i}\right)ln\sigma^2+nKln\sigma_p^2+mKln\sigma_q^2\right)+C,$$

化简得：

$$L=\frac{1}{2}\sum_{u=1}^{n}\sum_{i=1}^{m}I_{u,i}||r_{u,i}-\mathbf{p}_u\mathbf{q}_i^T||^2+\frac{\lambda_p}{2}\sum_{u=1}^{n}||\mathbf{p}_u||^2+\frac{\lambda_q}{2}\sum_{i=1}^{m}||\mathbf{q}_i||^2$$

详细推导见：

https://zhuanlan.zhihu.com/p/34422451

2. 随机梯度下降法（SGD）

当 $\lambda_p=\lambda_q$ 时，PMF 目标函数就如下：

目标函数：

$$\min_{\mathbf{p},\mathbf{q}}\frac{1}{2}\sum_{(u,i)\in\mathbf{O}}\left\|r_{u,i}-\mathbf{p}_u\mathbf{q}_i^T\right\|^2+\frac{1}{2}\lambda\left(\left\|\mathbf{p}_u\right\|^2+\left\|\mathbf{q}_i\right\|^2\right)$$

目标函数 L 分别对 $\mathbf{p}_u$ 和 $\mathbf{q}_i$ 进行求导得：

$$\frac{\partial L}{\partial\mathbf{q}_i}=-\left(r_{u,i}-\mathbf{p}_u\mathbf{q}_i^T\right)\mathbf{p}_u+\lambda\mathbf{q}_i$$

$$\frac{\partial L}{\partial \mathbf{p}_u} = -\left(r_{u,i} - \mathbf{p}_u \mathbf{q}_i^T\right)\mathbf{q}_i + \lambda \mathbf{p}_u$$

采用的是随机梯度下降法（SGD）进行求解，更新 $\mathbf{p}_u$ 和 $\mathbf{q}_i$ ：

$$\mathbf{p}_u \leftarrow \mathbf{p}_u - \eta\frac{\partial L}{\partial \mathbf{p}_u} = \mathbf{p}_u + \eta\left(\left(r_{u,i} - \mathbf{p}_u \mathbf{q}_i^T\right)\mathbf{q}_i - \lambda \mathbf{p}_u\right)$$

$$\mathbf{q}_i \leftarrow \mathbf{q}_i - \eta\frac{\partial L}{\partial \mathbf{q}_i} = \mathbf{q}_i + \eta\left(\left(r_{u,i} - \mathbf{p}_u \mathbf{q}_i^T\right)\mathbf{p}_u - \lambda \mathbf{q}_i\right)$$

令 $e_{ui} = r_{u,i} - \mathbf{p}_u \mathbf{q}_i^T$ ，上述式子简化为：

$$\mathbf{p}_u \leftarrow \mathbf{p}_u + \eta\left(e_{ui}\mathbf{q}_i - \lambda \mathbf{p}_u\right)$$

$$\mathbf{q}_i \leftarrow \mathbf{q}_i + \eta\left(e_{ui}\mathbf{p}_u - \lambda \mathbf{q}_i\right)$$

程序地址：

https://github.com/XiuzeZhou/Recommender-Systems/blob/master/mf.py

核心代码：

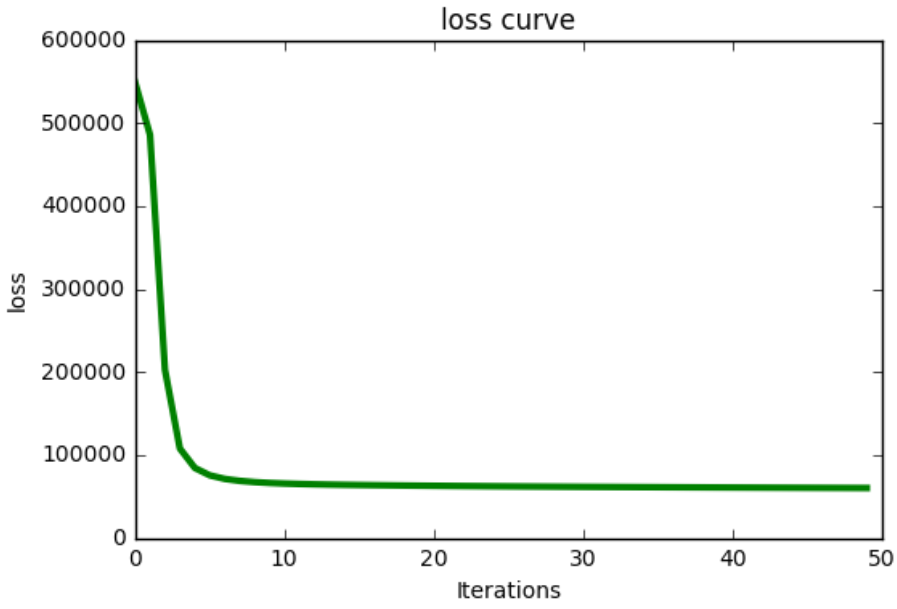
```
def update(p, q, r, learning_rate=0.001, lamda_regularizer=0.1):
    error = r - np.dot(p, q.T)
    p = p + learning_rate*(error*q - lamda_regularizer*p)
    q = q + learning_rate*(error*p - lamda_regularizer*q)
    loss = 0.5 * (error**2 + lamda_regularizer*(np.square(p).sum() +
np.square(q).sum()))
    return p,q,loss
```

实验结果：

数据集：Movielens100K，随机分割成训练集：测试集=8:2

MAE	RMSE	Recall@10	Precision@10
0.7347	0.9297	0.0293	0.0620

损失函数曲线：



3. 改进

1). 带偏置的SVD（ BiasSVD ）

来源出处：

- Koren et al. Matrix factorization techniques for recommender systems.Computer 42.8 (2009).

目标函数：

$$\min_{\mathbf{p},\mathbf{q}} \frac{1}{2} \sum_{(u,i)\in\mathbf{O}} \|r_{u,i} - \hat{r}_{u,i}\|^2 + \frac{1}{2}\lambda \left(\|\mathbf{p}_u\|^2 + \|\mathbf{q}_i\|^2 + \|b_u\|^2 + \|b_i\|^2 \right)$$

$$\hat{r}_{u,i} = \mu + b_u + b_i + \mathbf{p}_u \mathbf{q}_i^T$$

μ ：全部评分的均值

b_u ：用户 u 的评分均值

b_i ：商品 i 的评分均值

说明介绍：

该方法考虑了实际生活中，用户的评分偏好和商品的特性评分。比如，有对于某商品的好与不好，有用户评分很鲜明，给5和1分；有用户评分比较委婉，给5和3分。由此产生了不同的评分习惯。加入这些因素，用潜在特征来预测用户的喜好与"均值"的偏差更合理。

更新公式：

$$\mathbf{p}_u \leftarrow \mathbf{p}_u + \eta \left(e_{ui} \mathbf{q}_i - \lambda \mathbf{p}_u \right)$$

$$\mathbf{q}_i \leftarrow \mathbf{q}_i + \eta \left(e_{ui} \mathbf{p}_u - \lambda \mathbf{q}_i \right)$$

$$b_u \leftarrow b_u + \eta \left(e_{ui} - \lambda b_u \right)$$

$$b_i \leftarrow b_i + \eta \left(e_{ui} - \lambda b_i \right)$$

程序地址：

https://github.com/XiuzeZhou/Recommender-Systems/blob/master/biassvd.py

核心代码：

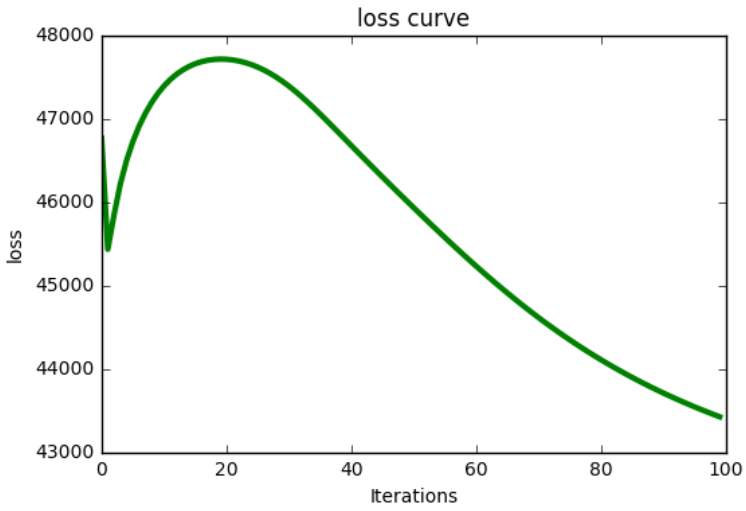
```
def update(p, q, bu, bi, aveg_rating, r, learning_rate=0.001,
lamda_regularizer=0.1):
    error = r - (aveg_rating + bu + bi + np.dot(p, q.T))
    p = p + learning_rate*(error*q - lamda_regularizer*p)
    q = q + learning_rate*(error*p - lamda_regularizer*q)
    bu = bu + learning_rate*(error - lamda_regularizer*bu)
    bi = bi + learning_rate*(error - lamda_regularizer*bi)
    return p,q,bu,bi
```

实验结果：

数据集：Movielens100K，随机分割成训练集：测试集=8:2

MAE	RMSE
0.7210	0.9124

loss 曲线：



这图是参数与本文其他模型相同时的收敛曲线，并不好看。

BiasSVD 的学习率不好调，调小 loss 曲线完美收敛，但是 MAE 和 RMSE 结果并不好看，应该是陷入了局部收敛区间；当调大时，loss 曲线又不好看，不过实验结果会好很多。我个人感觉是 BiasSVD 太“精细”了，反而容易陷入局部最优解。

2). SVD++

来源出处：

- Koren Y. Factor in the neighbors: Scalable and accurate collaborative filtering[J]. ACM Transactions on Knowledge Discovery from Data (TKDD), 2010, 4(1): 1.

目标函数：

$$\min \frac{1}{2} \sum_{(u,i)\in \mathbf{O}} \|r_{u,i} - \hat{r}_{u,i}\|^2 + \frac{1}{2}\lambda \left(\|\mathbf{p}_u\|^2 + \|\mathbf{q}_i\|^2 + \|b_u\|^2 + \|b_i\|^2 + \left\|\mathbf{y}_j\right\|^2 \right)$$

$$\hat{r}_{u,i} = \mu + b_u + b_i + \left(\mathbf{p}_u + |I_u|^{-\frac{1}{2}} \sum_{j\in I_u} y_j \right) \mathbf{q}_i^T$$

其中 I_u 为用户 u 评价过的所有电影的集合； $\mathbf{y}_j$ 为隐藏的对于商品 j 的隐含喜好； $|I_u|^{-\frac{1}{2}}$ 是一个经验公式。

说明介绍：

SVD++ 是 BiasSVD 的改进版，它考虑了用户的历史评分行为，将这些行为数据作为一个偏置加入到模型中，使模型更“精细”。

更新公式：

$$\mathbf{p}_u \leftarrow \mathbf{p}_u + \eta \left(e_{ui} \mathbf{q}_i - \lambda \mathbf{p}_u \right)$$

$$\mathbf{q}_i \leftarrow \mathbf{q}_i + \eta \left(e_{ui} \left(\mathbf{p}_u + |I_u|^{-\frac{1}{2}} \sum_{j\in I_u} \mathbf{y}_j \right) - \lambda \mathbf{q}_i \right)$$

$$b_u \leftarrow b_u + \eta\left(e_{ui} - \lambda b_u\right)$$

$$b_i \leftarrow b_i + \eta\left(e_{ui} - \lambda b_i\right)$$

$$\mathbf{y}_j \leftarrow \mathbf{y}_j + \eta\left(e_{ui}|I_u|^{-\frac{1}{2}}\mathbf{q}_i - \lambda \mathbf{y}_j\right)$$

程序地址：

https://github.com/XiuzeZhou/Recommender-Systems/blob/master/svdplus.py

核心代码：

```
def
update(p,q,bu,bi,Y,aveg_rating,r,Ru,learning_rate=0.001,lamda_regularizer=0.1):
    Iu = np.sum(Ru>0)
    y_sum = np.sum(Y[np.where(Ru>0)],axis=0)
    error = r - (aveg_rating + bu + bi + np.dot(p+Iu**(-0.5)*y_sum, q.T))

    p = p + learning_rate*(error*q - lamda_regularizer*p)
    q = q + learning_rate*(error*(p + Iu**(-0.5)*y_sum) - lamda_regularizer*q)
    bu = bu + learning_rate*(error - lamda_regularizer*bu)
    bi = bi + learning_rate*(error - lamda_regularizer*bi)

    for j in np.where(Ru>0):
        Y[j] = Y[j] + learning_rate*(error*Iu**(-0.5)*q -
lamda_regularizer*Y[j])

    return p,q,bu,bi,Y
```

实验结果：

数据集：Movielens100K，随机分割成训练集：测试集=8:2

MAE	RMSE
0.7162	0.9109

3). timeSVD

来源出处：

- Koren et al. Collaborative filtering with temporal dynamics. Communications of the ACM 53.4 (2010): 89-97.

目标函数：

$$\min \frac{1}{2}\sum_{(u,i)\in\mathbf{O}}\left\|r_{u,i} - \hat{r}_{u,i}\right\|^2 + \frac{1}{2}\lambda\left(\left\|\mathbf{p}_u\right\|^2 + \left\|\mathbf{q}_i\right\|^2 + \left\|b_u\right\|^2 + \left\|b_i\right\|^2\right)$$

$$\hat{r}_{u,i} = \mu + b_u\left(t\right) + b_i\left(t\right) + \mathbf{p}_u\left(t\right)\mathbf{q}_i^T$$

其中，t 为时间因子，表示不同的时间状态。

说明介绍：

文中假设：用户的兴趣是随时间变化的，即 $\mathbf{p}_u$ 与时间 t 相关。而 $\mathbf{q}_i$ 为商品的固有特征，与时间因素无关。比如，大部分用户夏天买短袖、短裤，冬天买长袖、羽绒服，时间效应明显。$\mathbf{q}_i$ 反映的是商品属性：你评价或者不评价，我都在这里，不增不减。同时，假设用户和商品的评分偏置也随时间 t 的变化而变化。

4. 模型对比

算法对比：

https://github.com/XiuzeZhou/Recommender-Systems/blob/master/MF%20Family.ipynb

在相同学习率 η、相同正则项系数 λ、相同特征维度 K、相同迭代次数的情况下，

即 learning_rate = 0.005，lamda_regularizer = 0.1，K = 10，max_iteration = 100

	MAE (比前一个算法提升 %)	RMSE (比前一个算法提升 %)
MF, SVD, Funk-SVD, PMF	0.7279 (-)	0.9297 (-)
BiasSVD	0.7203 (+1.0%)	0.9154 (+0.7%)
SVD++	0.7162 (+0.5%)	0.9109 (+0.5%)

从上到下，算法刚开始提升效果非常明显，到后来提升效果越来越小。当然，如果再调整参数，结果肯定还会有所提升。

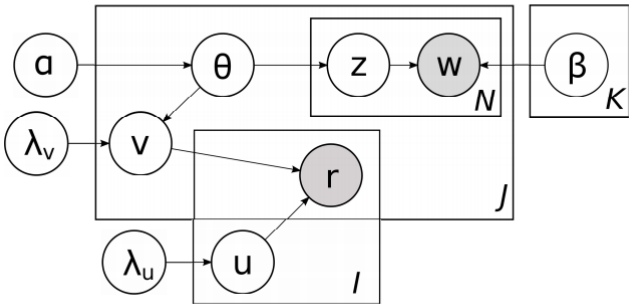
5. 拓展

1) 与主题模型结合

来源出处：

- Wang, Chong, and David M. Blei. "Collaborative topic modeling for recommending scientific articles." Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. 2011.

示意图：



大的框架为 LDA 主题模型，小的框架为 PMF 模型。

目标函数：

$$\min_{\mathbf{U},\mathbf{V}} \frac{1}{2}\sum_{(i,j)\in\mathbf{O}} c_{i,j}\left\|r_{u,i}-u_i v_j^T\right\|^2+\frac{1}{2}\lambda_u\sum_i\|u_i\|^2+\frac{1}{2}\lambda_v\sum_j\|v_j-\theta_j\|^2-\sum_j\sum_h log\left(\sum_k \theta_{i,k}\beta_{k,w_{j,h}}\right)$$

其中，θ_j 表示商品 j 文本信息的主题分布，$\beta_{k,w_{j,h}}$ 表示文中 j 中主题 k 下词语 h 的分布。

简单说明：

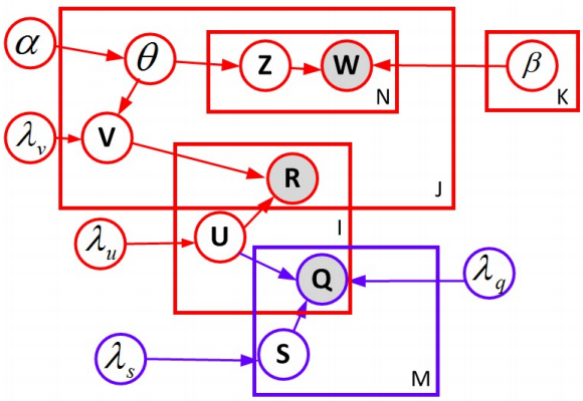
有些领域它们除了有交互信息外，文本内容比较丰富，比如新闻，学术论文。用主题模型获取文本信息，弥补交互信息不足时的情况。当交互信息丰富时，PMF 依旧其主要作用。

2) 与社交网络结合

来源出处：

- Purushotham, Sanjay, Yan Liu, and C-C. Jay Kuo. "Collaborative topic regression with social matrix factorization for recommendation systems." arXiv preprint arXiv:1206.4684 (2012).

示意图：



红色的框架为上一个模型，蓝色图为添加的社交信息框架。

目标函数：

$$\min_{\mathbf{U},\mathbf{V}} \frac{1}{2}\sum_{(i,j)\in\mathbf{O}} c_{i,j}\left\|r_{u,i}-u_i v_j^T\right\|^2+\frac{1}{2}\lambda_u\sum_i\|u_i\|^2+\frac{1}{2}\lambda_v\sum_j\|v_j-\theta_j\|^2-\sum_j\sum_h log\left(\sum_k \theta_{i,k}\beta_{k,w_{j,h}}\right)$$

$$+\frac{1}{2}\lambda_g\sum_{i,f}\left\|g_{i,f}-u_i s_f^T\right\|^2+\frac{1}{2}\lambda_s\sum_k\|s_k\|^2$$

其中，s_k 为用户的社交矩阵。

简单说明：

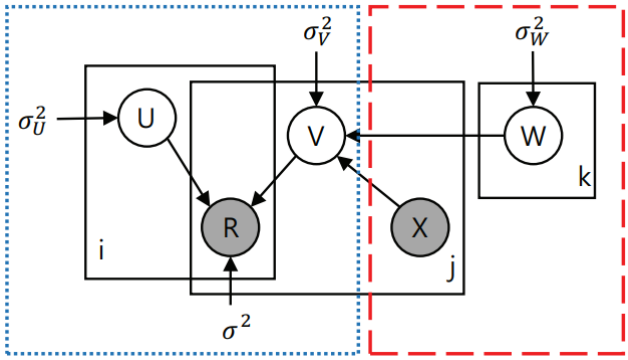
上一个模型上“丰富”了商品的特征向量，这个模型采用用户的社交信息来“丰富”用户的特征矩阵。

3) 与神经网络（CNN）结合

来源出处：

- Kim, Donghyun, et al. "Convolutional matrix factorization for document context-aware recommendation." Proceedings of the 10th ACM Conference on Recommender Systems. 2016.

示意图：



左侧为 PMF 模型，右侧为 CNN 构建。

目标函数：

$$\min_{\mathbf{U},\mathbf{V}} \frac{1}{2} \sum_{(i,j)\in\mathbf{O}} \left\| r_{u,i} - u_i v_j^T \right\|^2 + \frac{1}{2}\lambda_u \sum_i \|u_i\|^2 + \frac{1}{2}\lambda_v \sum_j \|v_j - cnn\left(W, X_j\right)\|^2 + \frac{1}{2}\lambda_k \sum_k \|w_k\|^2$$

其中，X_i 为商品 i 的文本评论，W 为 CNN 网络权重。

简单说明：

自从深度学习火了之后，很快就将深度学习的各种模型带入原来的 PMF 构架，来弥补 PMF 本身存在的不足。这里是是用 CNN 得到评论特征，丰富原来来自评分矩阵的特征向量。