推荐系统系列之二：矩阵分解家族

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. 非负矩阵分解（NMF）

来源出处：

- Lee et al. Learning the parts of objects by non-negative matrix factorization. Nature (1999): 788.

目标函数：

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

$$s.t. \, \mathbf{p}_{u,\cdot} > 0, \mathbf{q}_{i,\cdot} > 0$$

说明介绍：

在数学上，分解结果中存在负值是正确的。但在实际问题中，负值元素往往没有实际意义（如图像、文档）。

主要公式：

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

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

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

- **方法一**：随机梯度下降法（SGD）

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

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

其中 η 为步长，或者称之为学习率。

注意：下降的步长大小非常重要，因为如果太小，则找到函数最小值的速度就很慢，如果太大，则又可能会出现震荡。

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

$$\mathbf{p}_u \leftarrow \mathbf{p}_u + \eta \cdot e_{ui}\mathbf{q}_i$$

$$\mathbf{q}_i \leftarrow \mathbf{q}_i + \eta \cdot e_{ui}\mathbf{p}_u$$

因为 $\mathbf{p}_{u,\cdot} > 0, \mathbf{q}_{i,\cdot} > 0$ ，所以最终的更新公式：

$$\mathbf{p}_u \leftarrow max\left(0, \mathbf{p}_u + \eta \cdot e_{ui}\mathbf{q}_i\right)$$

$$\mathbf{q}_i \leftarrow max\left(0, \mathbf{q}_i + \eta \cdot e_{ui}\mathbf{p}_u\right)$$

- **方法二**：乘法迭代

来源：Daniel D. Lee 和 H. Sebastian Seung， "Algorithms for Non-negative Matrix Factorization"

$$\mathbf{p}_u \leftarrow \mathbf{p}_u + \eta_u \left(r_{u,i} - \mathbf{p}_u\mathbf{q}_i^T\right)\mathbf{q}_i$$

$$\mathbf{q}_i \leftarrow \mathbf{q}_i + \eta_i \left(r_{u,i} - \mathbf{p}_u\mathbf{q}_i^T\right)\mathbf{p}_u$$

改为：

$$\mathbf{P} \leftarrow \mathbf{P} + \eta_u \left(\mathbf{R} - \mathbf{P}\mathbf{Q}^T\right)\mathbf{Q} = \mathbf{P} + \eta_u \left(\mathbf{R}\mathbf{Q} - \mathbf{P}\mathbf{Q}^T\mathbf{Q}\right)$$

$$\mathbf{Q} \leftarrow \mathbf{Q} + \eta_i \left(\mathbf{R} - \mathbf{P}\mathbf{Q}^T\right)^T\mathbf{P} = \mathbf{Q} + \eta_i \left(\mathbf{R}^T\mathbf{P} - \mathbf{Q}\mathbf{P}^T\mathbf{P}\right)$$

令:

$$\eta_u = \frac{\mathbf{P}}{\mathbf{P}\mathbf{Q}^T\mathbf{Q}}$$

$$\eta_u = \frac{\mathbf{Q}}{\mathbf{Q}\mathbf{P}^T\mathbf{P}}$$

最终更新公式:

$$\mathbf{P} \leftarrow \mathbf{P} \cdot \frac{\mathbf{R}\mathbf{Q}}{\mathbf{P}\mathbf{Q}^T\mathbf{Q}}$$

$$\mathbf{Q} \leftarrow \mathbf{Q} \cdot \frac{\mathbf{R}^T\mathbf{P}}{\mathbf{Q}\mathbf{P}^T\mathbf{P}}$$

程序地址：

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

核心代码：

- 方法一：

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

- 方法二：

```
def update(P, Q, R ,eps=1e-6):
    P = P * (np.dot(R+eps,Q)/(np.dot(P,np.dot(Q.T,Q)))+eps)
    Q = Q * (np.dot(R.T+eps,P)/(np.dot(Q,np.dot(P.T,P)))+eps)
    return P, Q
```

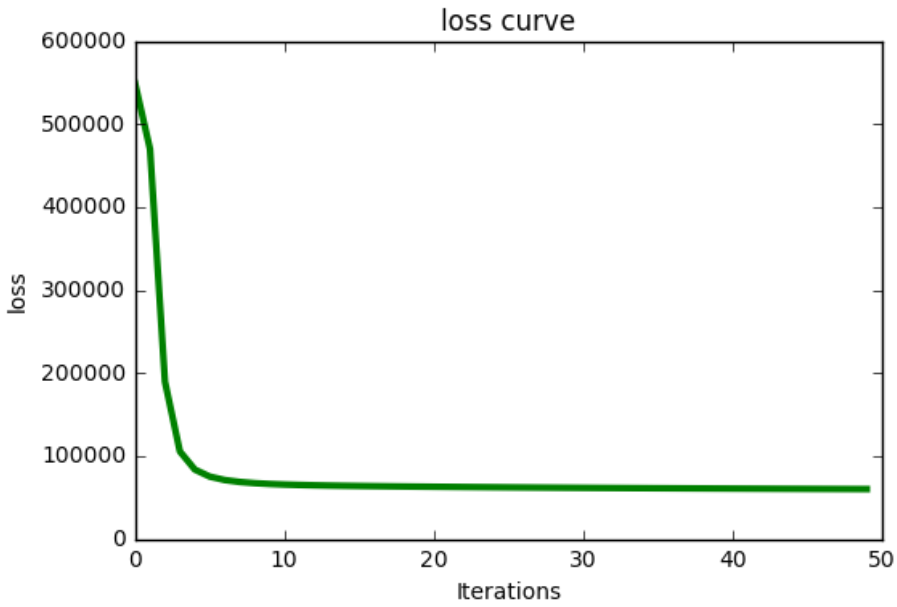
实验结果：

数据集：Movielens100K，随机分割成训练集：测试集=8:2

- 方法一：

MAE	RMSE
0.7623	0.9653

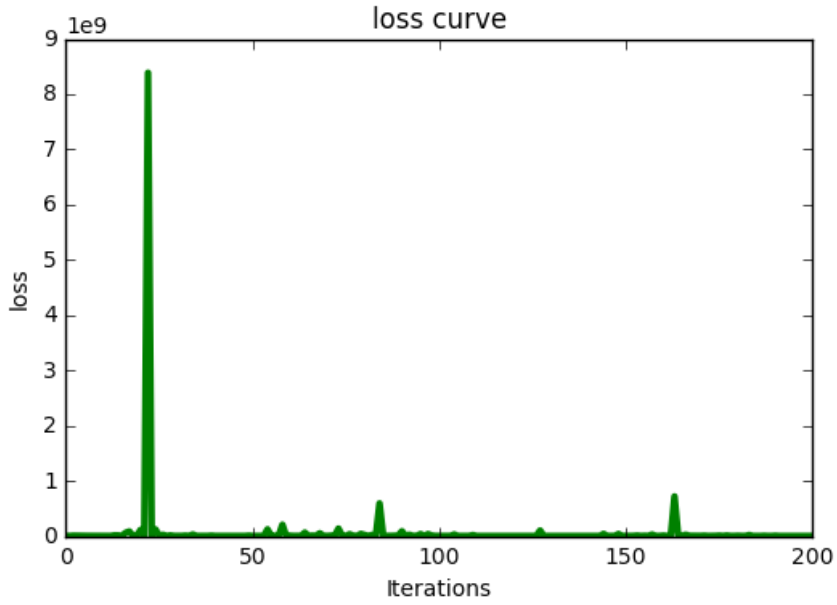
损失函数曲线：



- 方法二：

MAE	RMSE
2.7498	9.8933

损失函数曲线：



- sklearn 自带的 NMF 函数实验结果：

MAE	RMSE
2.3646	6.9440

总结：

个人认为乘法迭代表现不佳的原因，可能是输入数据是稀疏，更新公式中 $\mathbf{RQ}$ 只有少量非零项叠加，从而使得通过计算得到的步长太大，每次迭代更新幅度太大。

2. 矩阵分解（MF，SVD，Funk-SVD）

来源出处：

- http://sifter.org/-simon/.
- Koren et al. Matrix factorization techniques for recommender systems.Computer 42.8 (2009).

目标函数：

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

说明介绍：

没有 NMF 公式中的非负的限制条件，且加入了 L2 正则项来防止过拟合。前者使模型更“精确”，后者使模型更“稳定”。这是最通用的模型，之后大部分基于矩阵分解的算法都是在此基础上改进而来。

主要公式：

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：

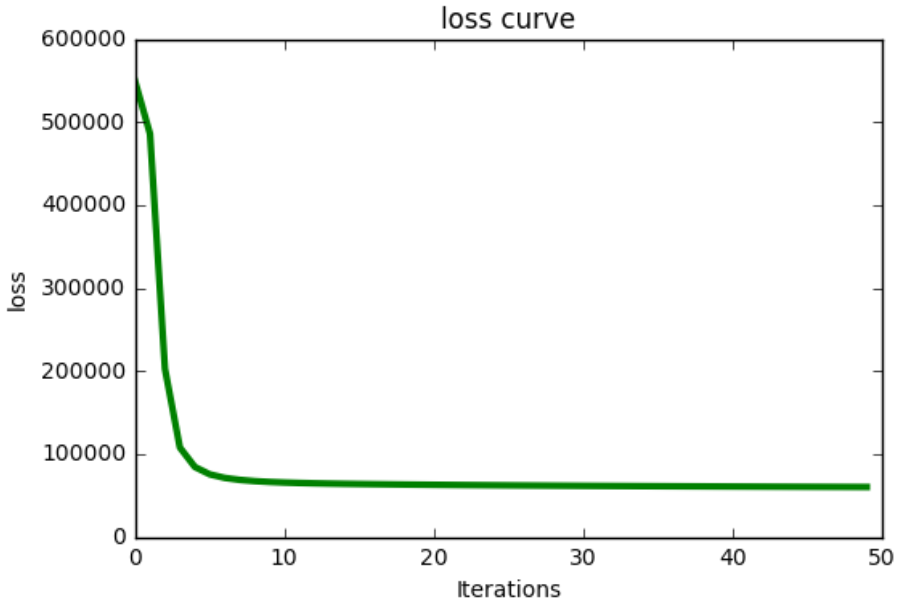
```
def update(p, q, r, learning_rate=0.001, lamda_regularizer=0.1):
    error = r - np.dot(p, q.T)
    p = p + learning_rate*(error*q - lamda_regularizer*p)
    q = q + learning_rate*(error*p - lamda_regularizer*q)
    loss = 0.5 * (error**2 + lamda_regularizer*(np.square(p).sum() +
np.square(q).sum()))
    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	0.8643	0.0293	0.0620

损失函数曲线：



3. 概率矩阵分解 （ PMF ）

来源出处：

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

目标函数：

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

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

当 $\lambda_p=\lambda_q$ 时，PMF 目标函数就与之前的 MF 一样。

说明介绍：

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

主要公式：

假设关于已知评分数据的条件分布满足高斯分布：

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

其中，$\mathbf{I}_{u,i}$ 为指示函数：如果用户 u 已经对商品 i 进行了评分，则为1；否者为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：

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

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

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

等式两边取对数 ln 后得到：

$$\begin{aligned}lnp\left(\mathbf{p},\mathbf{q}|\mathbf{R},\sigma^2,\sigma_p^2,\sigma_q^2\right)=&-\frac{1}{2\sigma^2}\sum_{u=1}^{n}\sum_{i=1}^{m}\mathbf{I}_{ij}\left(r_{u,i}-\mathbf{p}_u\mathbf{q}_i^T\right)^2-\frac{1}{2\sigma_p^2}\sum_{i=1}^{N}\mathbf{p}_u\mathbf{p}_u^T-\frac{1}{2\sigma_q^2}\sum_{i=1}^{M}\mathbf{q}_i\mathbf{q}_i^T\\&-\frac{1}{2}\left(\left(\sum_{i=1}^{n}\sum_{j=1}^{m}I_{u,i}\right)ln\sigma^2+nKln\sigma_p^2+mKln\sigma_q^2\right)+C,\end{aligned}$$

详细推导见：

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

程序地址：

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

核心代码：与 MF 一样

```
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：与 MF 一样

数据集：Movielens100K，随机分割成训练集：测试集=8:2

MAE	RMSE
0.7347	0.8643

4. 权重矩阵分解（WMF）

来源出处：

- Pan et al. One-class collaborative filtering. ICDM, 2008.
- Hu et al. Collaborative filtering for implicit feedback datasets. ICDM, 2008.

目标函数：

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

$$w_{u,i} = \begin{cases} 1 \\ \delta \in (0,1) \end{cases}$$

或者

$$w_{u,i} = 1 + \alpha r_{u,i}$$

说明介绍：

在隐式反馈（如点击、浏览网页等行为）的数据中，只有正样本，而没有负样本。为了解决这一问题，提出对正样本和未知样本加以不同的权重以区分不同样本的重要程度。在 MF 中，对待正样本和未知样本的“态度”是一样的，即 $w_{u,i}$ 都是1。

更新公式：

$$p_u \leftarrow \left(Q^T C^u Q + \lambda I\right)^{-1} Q^T C^u R(u)$$

$$q_i \leftarrow \left(P^T C^i P + \lambda I\right)^{-1} P^T C^i R(i)$$

其中，$C^u \in \mathbb{R}^{n \times n}$ 且 $C^u_{i,i} = c_{u,i}$；$C^i \in \mathbb{R}^{m \times m}$ 且 $C^i_{u,u} = c_{u,i}$；$R(u)$ 表示用户-商品矩阵中用户 u 的所在向量；$R(i)$ 表示商品 i 的所在向量。

程序地址：

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

核心代码：

```
def update(P, Ru, lamda_regularizer=0.1, alpha=40):
    # P: N/M *K
    # Ru: N/M *1
    N, K = P.shape
    c_ui = 1 + alpha*Ru
    Cu = c_ui* np.eye(N)

    YtCY_I = P.T.dot(Cu).dot(P) + lamda_regularizer*np.eye(K)
    YtCRu = P.T.dot(Cu).dot(Ru)
    p = np.linalg.inv(YtCY_I).dot(YtCRu)
    return p.T
```

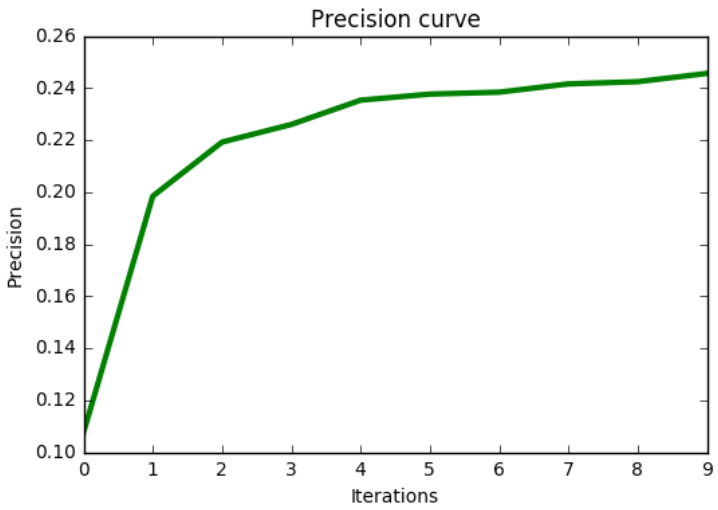
实验结果：

数据集：Movielens100K，随机分割成训练集：测试集=8:2

当 lamda_regularizer = 500，K = 100 时，即原论文中的参数设置。

Recall@10	Precision@10
0.1158	0.2456

Precision 曲线：



当 lamda_regularizer = 0.1，K = 10 时，

Recall@10	Precision@10
0.0793	0.1681

该组实验结果用来和本文中其他算法做对比。

5. 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：

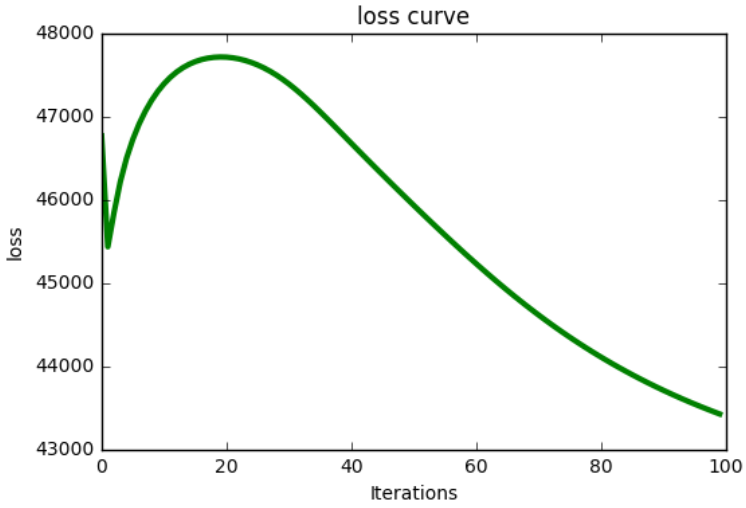
```
def update(p, q, bu, bi, aveg_rating, r, learning_rate=0.001,
lamda_regularizer=0.1):
    error = r - (aveg_rating + bu + bi + np.dot(p, q.T))
    p = p + learning_rate*(error*q - lamda_regularizer*p)
    q = q + learning_rate*(error*p - lamda_regularizer*q)
    bu = bu + learning_rate*(error - lamda_regularizer*bu)
    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	0.8325

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。

6. 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：

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

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

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
0.7197	0.8341

7. 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。

8. PureSVD

来源出处：

- Cremonesi et al. Performance of Recommender Algorithms on Top-N Recommendation Tasks. RecSys2010

主要公式：

$$\widehat{\mathbf{R}} = \mathbf{U} \cdot \mathbf{\Sigma} \cdot \mathbf{Q}^T$$

$$\mathbf{P} = \mathbf{U} \cdot \mathbf{\Sigma}$$

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

此处，$\hat{r}_{u,i}$ 为用户 u 对商品 i 的分数（score），而非评分（rating）。

说明介绍：

对评分矩阵进行填充后，再采用 SVD 分解。然后根据分解后的向量来预测分值，而非评分，用于排序筛选出 topn 列表。

程序地址：

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

核心代码：

```
def puresvd(R = None, #训练集
            k=150, #潜在因素
            ):
    P, sigma, QT = randomized_svd(R, k)
    sigma = scipy.sparse.diags(sigma, 0)
    P = P * sigma
    Q = QT.T
    # R_= np.dot(P, QT)
    R_ = np.dot(R, np.dot(Q, QT)) #
    return R_
```

实验结果：

数据集：Movielens100K，随机分割成训练集：测试集=8:2

Recall@10	Precision@10
0.1629	0.3455

9. 以后有空再补充...

10. 总结

算法对比：

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

1) 精度指标

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 %)
NMF	0.7814 (-)	1.0266 (-)
MF, **SVD**, **Funk-SVD**, **PMF**	0.7279 (+6.8%)	0.8517 (+17.0%)
BiasSVD	0.7203 (+1.0%)	0.8379 (+1.6%)
SVD++	0.7162 (+0.5%)	0.8297 (+1.0%)

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。

2) 排序指标

在相同正则项系数 λ、相同特征维度 K 的情况下，

即 lamda_regularizer = 0.1，K = 10

	Recall@10	Precision@10
MF, **SVD**, **Funk-SVD**, **PMF**	0.0271	0.0575
WMF	0.0780	0.1655
PureSVD	0.1618	0.3432

MF 系列算法 在 MAE 和 RMSE 上表现不错，毕竟它的目标函数就是平方差，也可以看成 RMSE。但是，它们在排序指标上表现并不理想。PureSVD 这个简单粗暴的方法，实验结果意外得很好！