

5.4 基于模型的协同过滤推荐

CSDN学院
2017年11月



► 大纲

- 推荐系统出现的背景
- 基于协同过滤的推荐
 - 基于用户的协同过滤
 - 基于物品的协同过滤
 - 基于模型的协同过滤
- 基于内容的推荐
- 推荐系统的评价
- 案例分析

▶ 基于模型的协同过滤

- 基本思想：离线训练一个比较复杂的模型，但在线预测时快速
- 可用类似奇异值分解（ Singular Value Decomposition , SVD ）对评分矩阵R进行降维（亦被称为隐因子分解LFM）
 - 抓住重要的因素（隐含因子）及其权重
 - 这些重要因素可能是人能理解的（如电影:演员、题材、主题、年代...）
 - 也可能是人不容易理解的因素
 - 假设重要因素的维数为 K ($K = 20$ to 100)
- 预测所需的时间为常数

► Recall : SVD分解

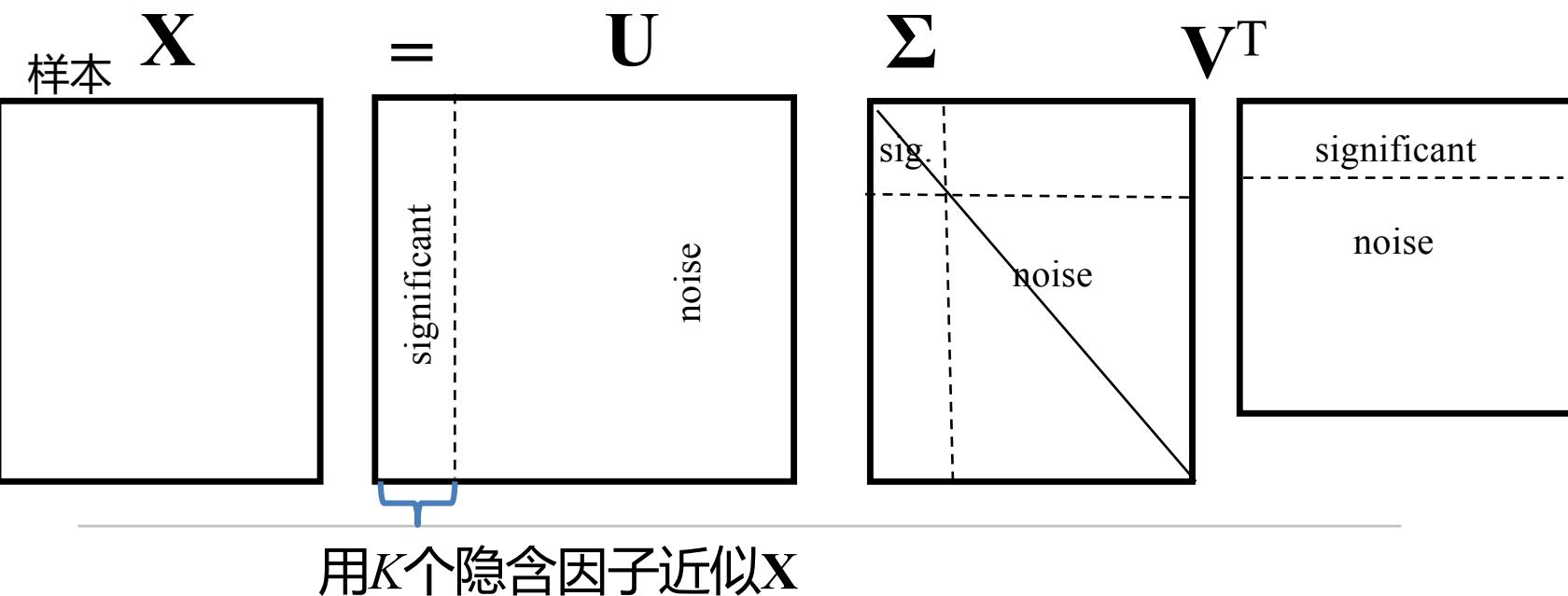
左奇异向量

奇异值矩阵

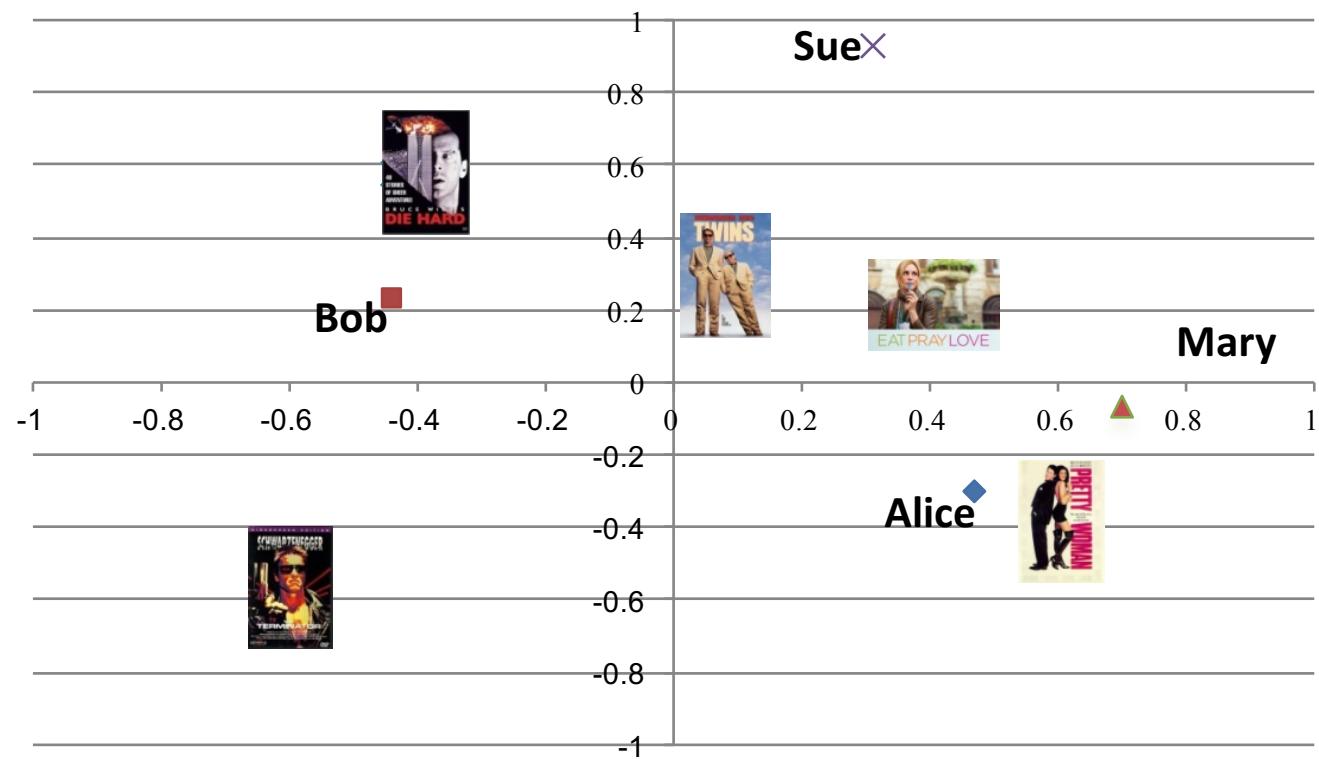
右奇异向量

$$\mathbf{X}_{N \times D} = \mathbf{U}_{N \times N} \mathbf{S}_{N \times D} \mathbf{V}^T_{D \times D}$$

- SVD : 将一个复杂的矩阵用更简单的3个子矩阵的相乘表示
 - 应用于LSA(隐性语义分析)、推荐系统、数据降维



► 如图所示 ...



二维空间中用户对电影的评分

► 矩阵分解

- SVD: $M_k = U_k \times \Sigma_k \times V_k^T$

U_k	Dim1	Dim2
Alice	0.47	-0.30
Bob	-0.44	0.23
Mary	0.70	-0.06
Sue	0.31	0.93

V_k^T					
Dim1	-0.44	-0.57	0.06	0.38	0.57
Dim2	0.58	-0.66	0.26	0.18	-0.36

Σ_k	Dim1	Dim2
Dim1	5.63	0
Dim2	0	3.23

- Prediction: $\hat{r}_{ui} = \bar{r}_u + U_k(Alice) \times \Sigma_k \times V_k^T(EPL)$
 $= 3 + 0.84 = 3.84$

► 矩阵分解

- SVD分解看起来很好，但是
 - 时间复杂度是 $O(N^3)$
 - 同时评分矩阵R缺失值太多
- 因此最简单的办法是直接矩阵分解

$$R_{U \times I} = P_{U \times K} Q_{K \times I}$$

User /item	1	2	3	4	5
1	5	4	4.5	?	3.9
2	?	4.5	?	4.5	?
3	4.5	?	4.4	4	4
4	?	4.8	?	?	4.5
5	4	?	4.5	5	?
.....

► 矩阵分解

- 矩阵R: 用户对物品的打分
 - 行：用户user
 - 列：物品item
- 假定有 U 个用户， I 个item， K 个隐含变量，我们需要找到矩阵P和Q，使得

$$R_{U \times I} = P_{U \times K} Q_{K \times I}$$

- 那么未知的评分为: $\hat{r}_{ui} = p_u^T q_i$

User /item	1	2	3	4	5
1	5	4	4.5	?	3.9
2	?	4.5	?	4.5	?
3	4.5	?	4.4	4	4
4	?	4.8	?	?	4.5
5	4	?	4.5	5	?
.....

<http://blog.csdn.net/zongkejingwang>

► 模型训练

- 利用R中的已知评分，训练P和Q，使得P和Q相乘的结果能最好地拟合已知的评分
- 目标函数为最小化总的误差平方和

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

- 模型求解：梯度下降

► 梯度下降

- 目标函数：

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

- 梯度下降：

偏导： $\frac{\partial}{\partial p_{uk}} SSE = -e_{ui} q_{ki}$, $\frac{\partial}{\partial q_{ki}} SSE = -e_{ui} p_{uk}$

更新公式： $p_{uk} := p_{uk} - \eta (-e_{ui} q_{ki}) := p_{uk} + \eta e_{ui} q_{ki}$

————— $q_{ki} := q_{ki} - \eta (-e_{ui} p_{uk}) := q_{ki} + \eta e_{ui} p_{uk}$

► 增加正则项(RSVD)

- 上述模型是考虑了训练集上的误差，可能会导致拟合
- 因此增加正则项，目标函数变为：

$$\begin{aligned} SSE &= \frac{1}{2} \sum_{u,i} e_{ui}^2 + \frac{1}{2} \lambda \sum_u |p_u|^2 + \frac{1}{2} \lambda \sum_i |q_i|^2 \\ &= \frac{1}{2} \sum_{u,i} e_{ui}^2 + \frac{1}{2} \lambda \sum_u \sum_{k=0}^K p_{uk}^2 + \frac{1}{2} \lambda \sum_i \sum_{k=0}^K q_{ki}^2 \end{aligned}$$

- 导数变为： $\frac{\partial}{\partial p_{uk}} SSE = -e_{ui} q_{ki} + \lambda p_{uk}$

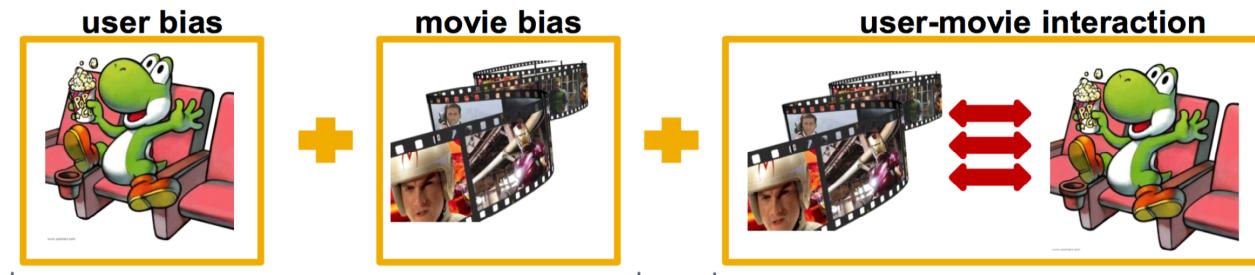
$$\frac{\partial}{\partial q_{ik}} SSE = -e_{ui} p_{uk} + \lambda q_{ik}$$

► 增加偏差项

- 用户对商品的打分不仅取决于用户和商品间的某种关系，
- 还取决于用户和商品独有的性质

$$\hat{r}_{ui} = \mu + b_u + b_i + p_u^T q_i$$

μ : 总的平均分
 b_u : 用户 u 的属性值
 b_i : 商品 i 的属性值



Baseline predictor

- Separates users and movies
- Benefits from insights into user's behavior
- Among the main practical contributions of the competition

User-Movie interaction

- Characterizes the matching between users and movies
- Attracts most research in the field
- Benefits from algorithmic and mathematical innovations

► 增加偏差项

- 目标函数变为

$$\begin{aligned} SSE &= \frac{1}{2} \sum_{u,i} e_{ui}^2 + \frac{1}{2} \lambda \sum_u |p_u|^2 + \frac{1}{2} \lambda \sum_i |q_i|^2 + \frac{1}{2} \lambda \sum_u b_u^2 + \frac{1}{2} \lambda \sum_i b_i^2 \\ &= \frac{1}{2} \sum_{u,i} (r_{ui} - \mu - b_u - b_i - \sum_{k=1}^K p_{uk} q_{ki})^2 + \frac{1}{2} \lambda \sum_u |p_u|^2 + \frac{1}{2} \lambda \sum_i |q_i|^2 + \frac{1}{2} \lambda \sum_u b_u^2 + \frac{1}{2} \lambda \sum_i b_i^2 \end{aligned}$$

- 导数：
 - SSE对 p_{uk} 和 q_{ik} 的导数都没有变化，但此时多了 b_u 和 b_i 变量

$$\frac{\partial}{\partial b_u} SSE = -e_{ui} + \lambda b_u \quad , \quad \frac{\partial}{\partial b_i} SSE = -e_{ui} + \lambda b_i$$

► SVD++：带历史隐式反馈的矩阵分解

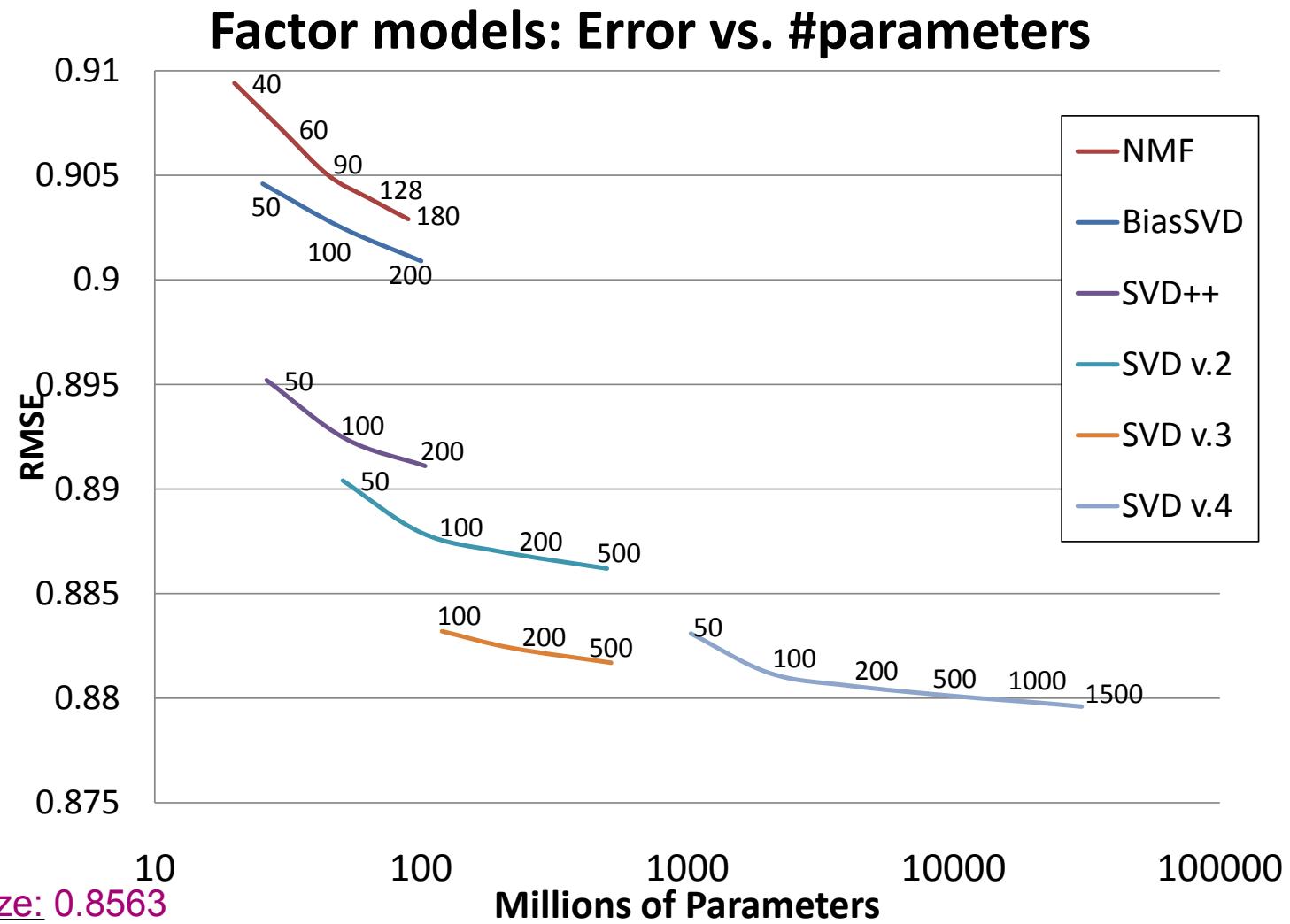
Koren在NetFlix大赛中用到的SVD算法

- 现实的评分矩阵特别稀疏, 为了使得数据更加稠密 , 加入隐式反馈数据
 - 如用户浏览过某个电影就可以当做一定的喜好的正反馈
- 隐式反馈表现的偏好: $|N(u)|^{-0.5} \sum_{j \in N(u)} x_j$
 - 其中 $N(u)$ 表示用户 u 评分/浏览过的物品
 - x_j 表示历史数据所表现出的偏好的向量
- 评分预测公式:

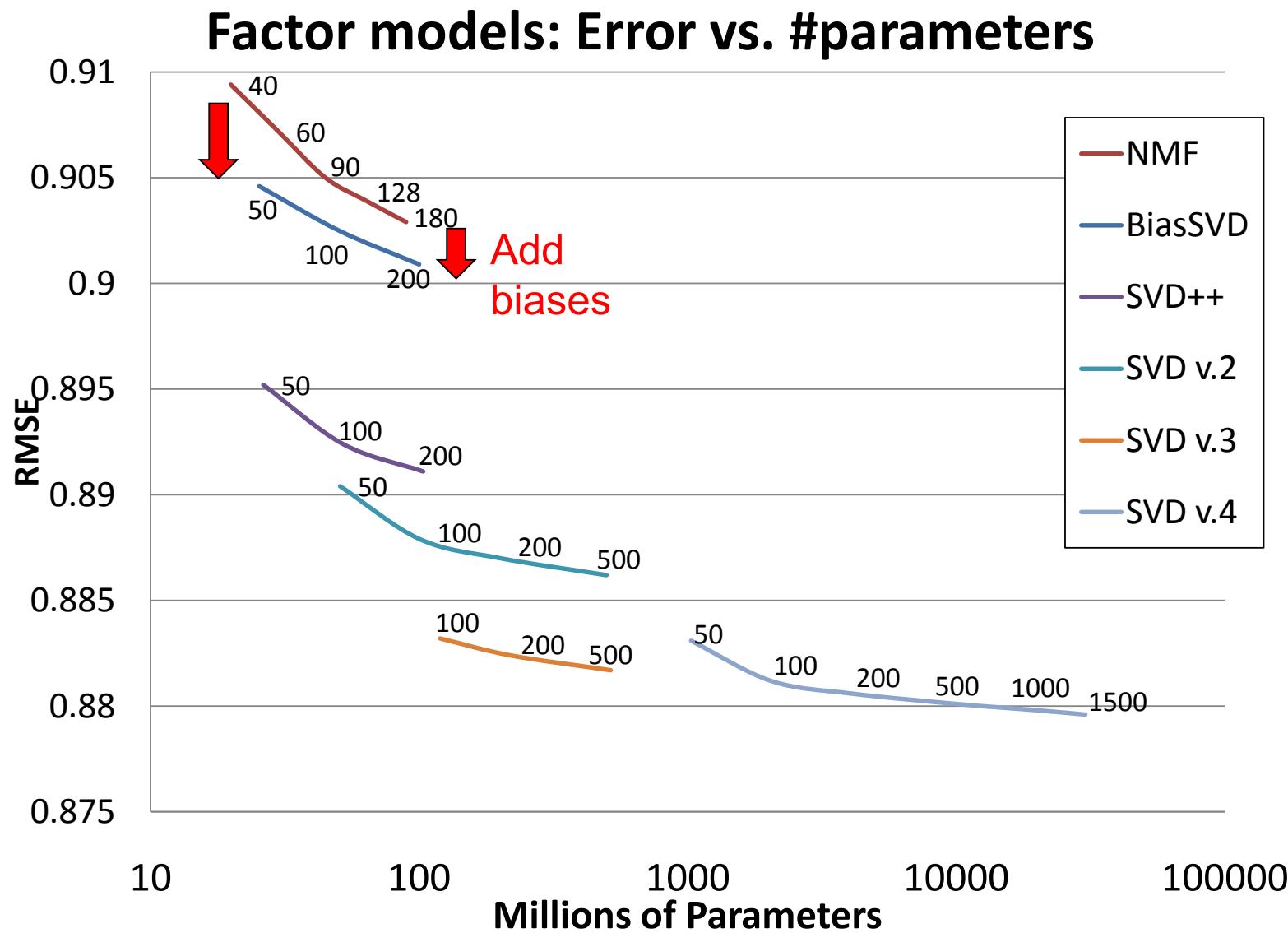
$$\hat{r}_{ui} = u + b_i + b_u + \mathbf{q}_i^T \left(\mathbf{p}_u + |N(u)|^{-0.5} \sum_{j \in N(u)} x_j \right)$$



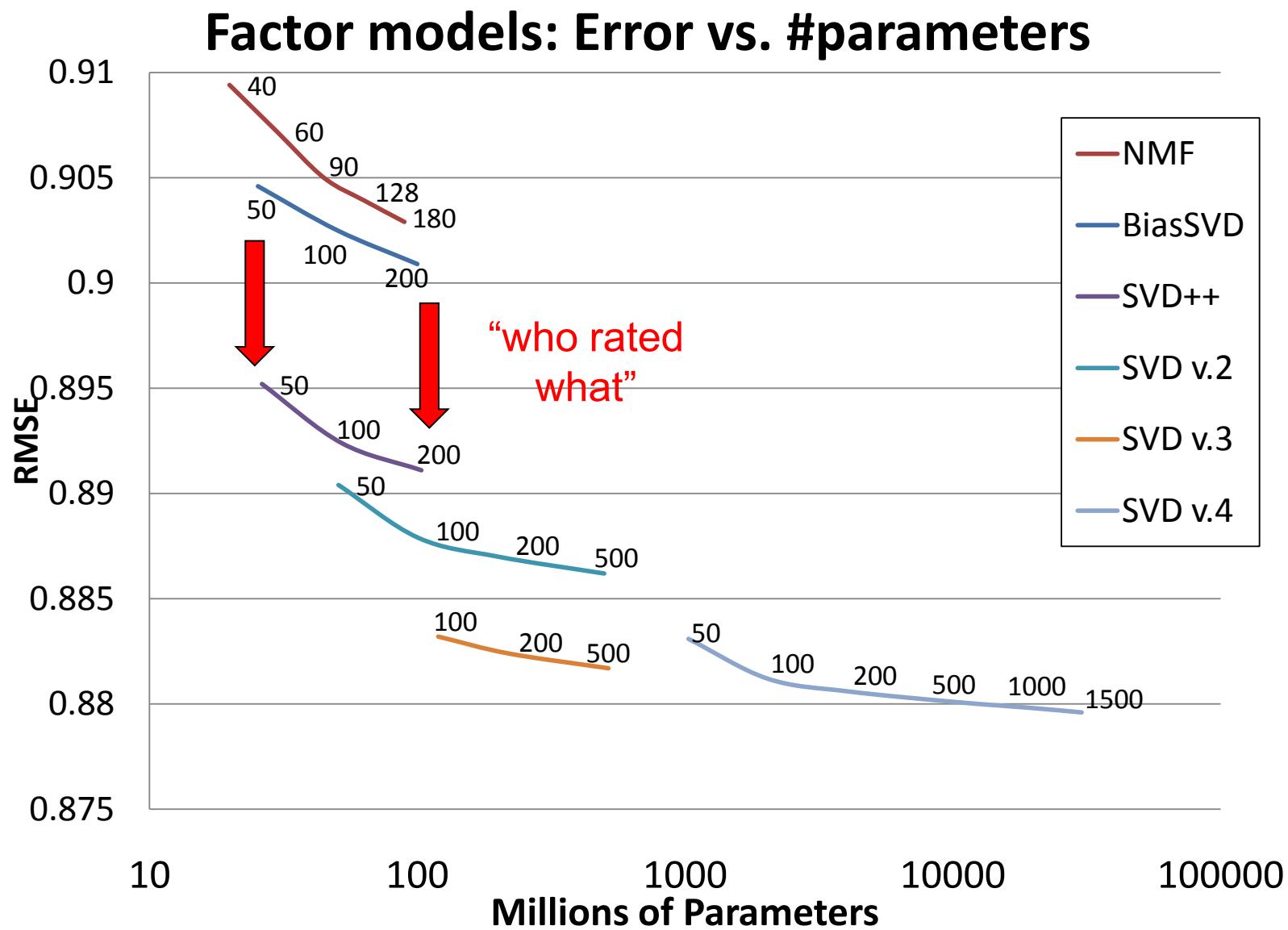
Netflix: 0.9514

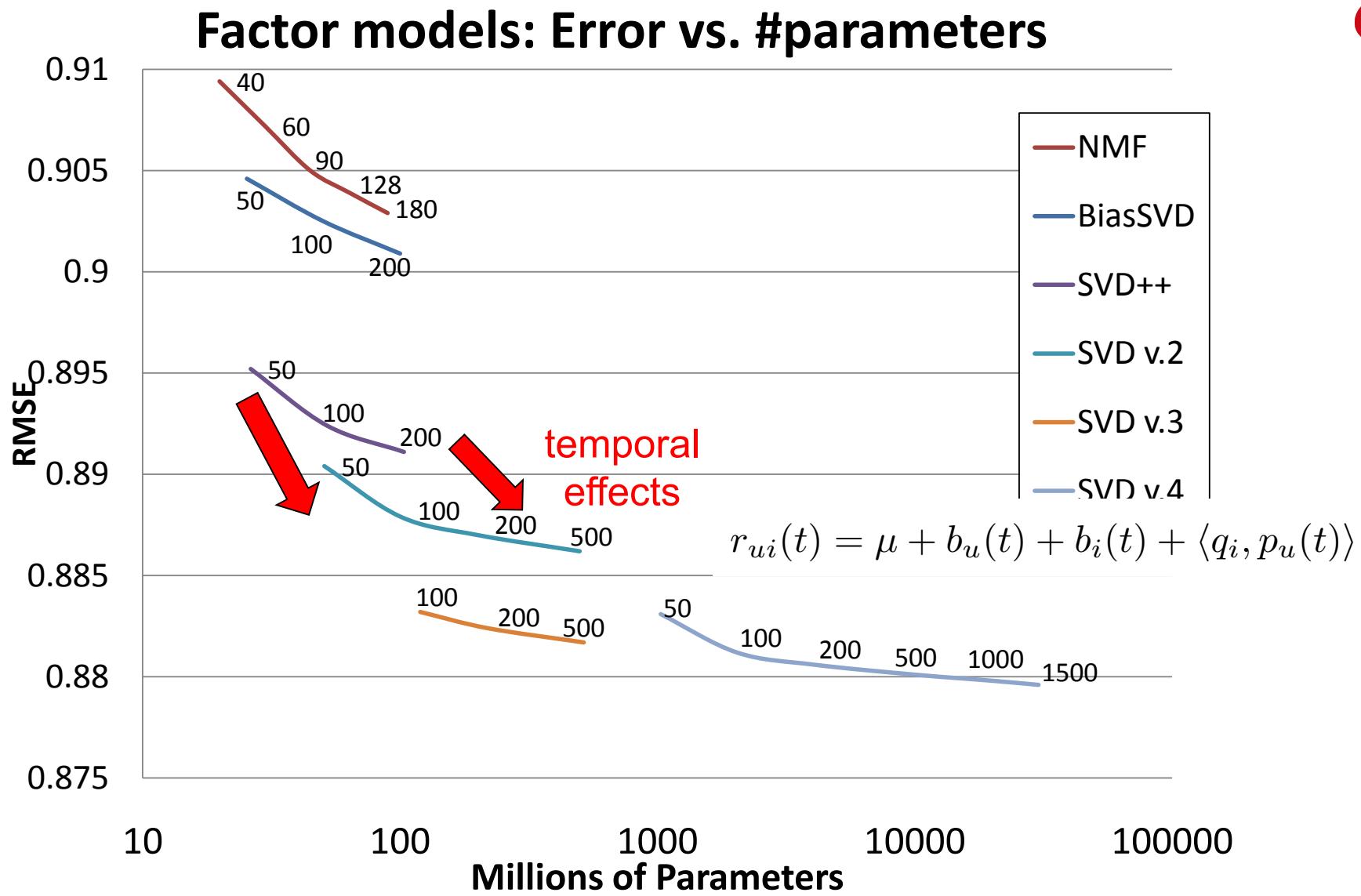


Prize: 0.8563

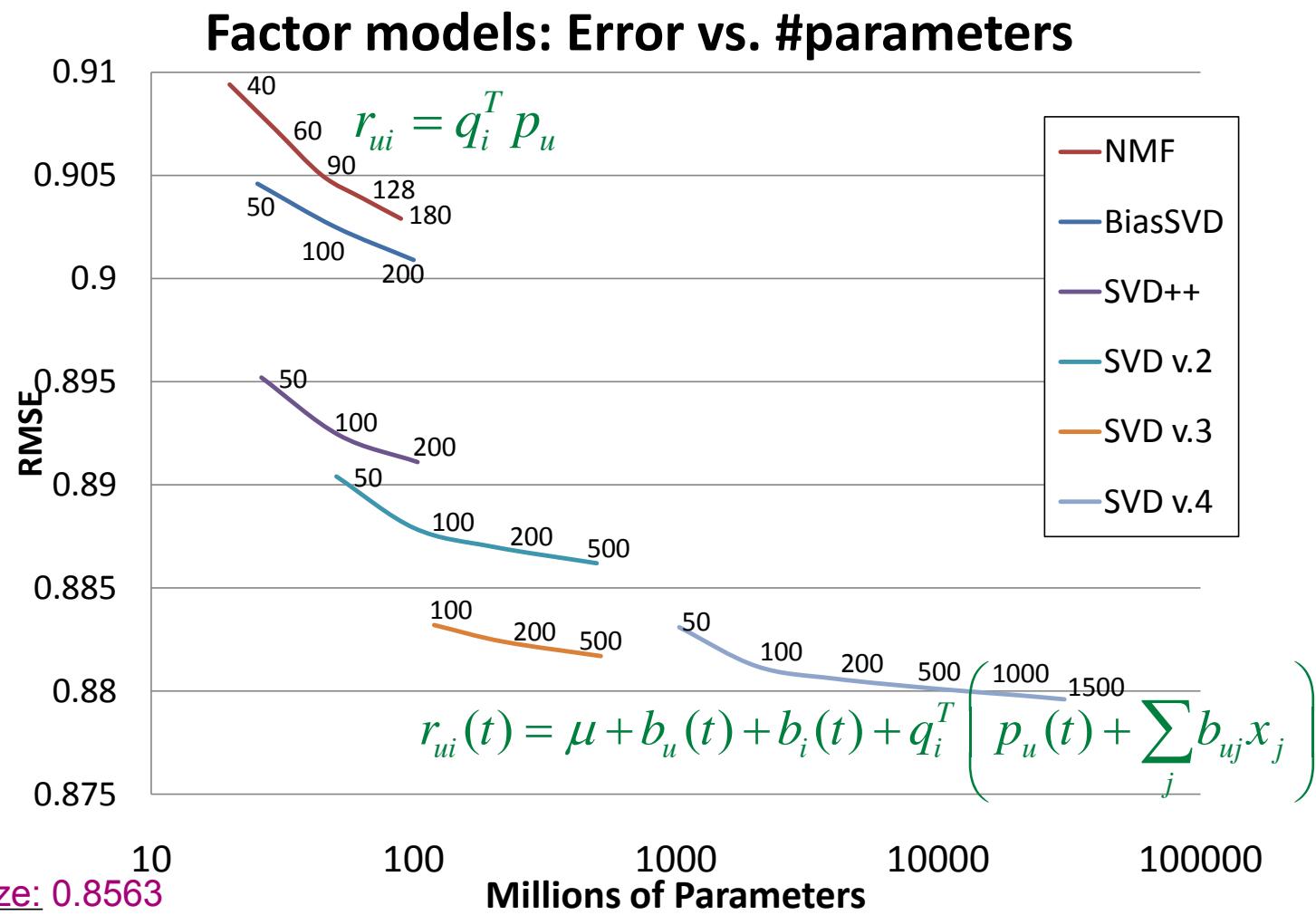


隐式反馈





Netflix: 0.9514



Prize: 0.8563

► 协同过滤优缺点

- 协同过滤优点
 - 基于用户行为，因此对推荐内容无需先验知识
 - 只需要用户和商品关联矩阵即可
 - 结构简单
 - 在用户行为丰富的情况下，效果好
- 协同过滤缺点
 - 需要大量的显性/隐性用户行为
 - 需要通过完全相同的商品关联（相似商品不行）
 - 假定用户的兴趣完全取决于之前的行为，和当前上下文环境无关
 - 在数据稀疏的情况下受影响(可考虑二度关联)

► 开源工具

- [SVDFeature](#) : 一个feature-based协同过滤和排序工具，由上海交大Apex实验室开发
 - 语言 : C++
 - 在KDD Cup 2012中获得第一名
 - 论文 发表在2012的JMLR
 - 包含Matrix Factorization推荐框架，能方便的实现SVD、 SVD++等方法
 - 代码精炼，可用相对较少的内存实现较大规模的单机版矩阵分解运算
 - 含有Logistic regression



不止于代码



THANK YOU



AI100

