Julia Tutorial Series - IV

Recommender System with Julia

Abhijith Chandraprabhu

February 1, 2014

Overview

What is Recommender System?

Recommender system is a subclass of Information Filtering, where it predicts how the constituent members of 2 different groups, unfamiliar with each other interact.

$$
\begin{array}{rccc}
 & GroupA & & GroupB \\
NETFLIX: & [Users] & \longrightarrow & [Movies] \\
Linkedin: & [Users] & \longrightarrow & [Connections], [Groups] \\
Amazon: & [Customers] & \longrightarrow & [Products] \\
Newspaper: & [Readers] & \longrightarrow & [Articles] \\
Coursera: & [Students] & \longrightarrow & [Courses]
\end{array}
$$

Types of Recommender Systems

Explicit				*Implicit*			
NETFLIX				*Amazon*			
	M1	M2	M3		P1	P2	P3
U1	5	?	2	U1	Yes	?	No
U2	?	?	4	U2	?	?	Yes
U3	4	1	?	U3	Yes	No	?

$Ratings : 5, 4, 3, 2, 1$ | $PurchaseHistory : Yes, No$

NETFLIX

- Netflix, Inc is a company based in USA, which provides internet streaming media on-demand.
- 33 million members view over 1 billion hours of TV shows and movies through NETFLIX per month.
- Personalized service through recommendations based on previous ratings.
- In October 2006, NETFLIX announced a prize of 1 Million USD to beat *Cinematch* by 10%
- ACM conference RecSys started from 2007.

Example: Let us consider a simple toy example of 4 users and 4 movies.

	Titanic	Braveheart	The Lion King	Dreamcatcher
John	5	5	2	2
Dave	2	?	3	4
Alice	4	5	?	3
Bob	3	4	2	5

where ? denotes the *user-movie* combinations which has to be predicted.

Dataset

$$\mathrm{R} = \begin{pmatrix} r_{11} & \cdots & r_{1n} \\ \vdots & \ddots & \vdots \\ r_{m1} & \cdots & r_{mn} \end{pmatrix} \qquad \mathrm{D} = \begin{pmatrix} d_{11} & \cdots & d_{1n} \\ \vdots & \ddots & \vdots \\ d_{m1} & \cdots & d_{mn} \end{pmatrix}$$

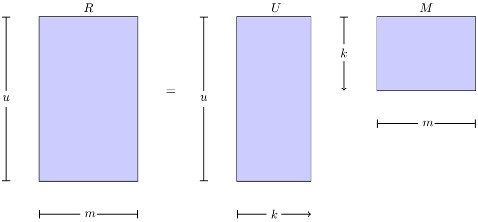
- The values of R are $\{r_{mn}{:}1 \leq \mathrm{r}_{mn} \leq 5 | \mathrm{r}_{mn} \text{ is an integer}\}$
- The values of D are $\{d_{mn}{:}1 \leq \mathrm{d}_{mn} \leq 2243 | \mathrm{d}_{mn} \text{ is an integer}\}$
- The values d_{mn} are in the matlabs Serial Date Number, which is offset from some starting date.

Training and Test datasets

- The *Training dataset*, consists of 99,077,112 ratings, plus 1,408,395 as probe set.
- *Qualifying dataset*, consists of 2,817,131 ratings, of which *test dataset* is 1,408,789 ratings, and *quiz dataset* is 1,408,342 ratings.

Problem Formulation

To perform matrix completion through low-rank approximation, in which the cost function measures the fit between a given matrix (the data) and an approximating matrix (the optimization variable), subject to a constraint that the approximating matrix has reduced rank.

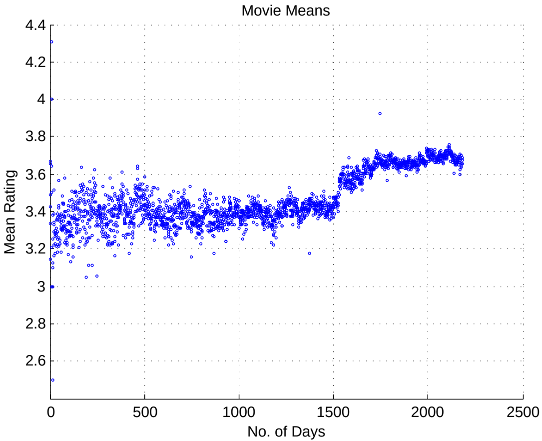


$$R \in \mathbb{R}^{u,m} \qquad U \in \mathbb{R}^{u,k} \qquad M \in \mathbb{R}^{m,k}$$

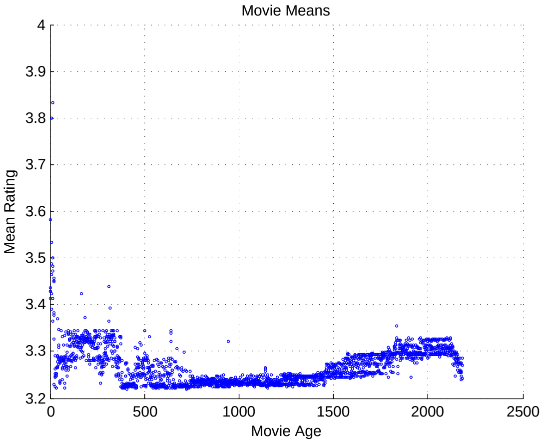
$$\hat{r}_{ij} = \mathbf{u}_i \cdot \mathbf{m}_j$$

- Matrix Factorization characterizes the users in U and movies in M.
- By mapping the users and movies into a joint latent factor space of k dimension.
- Singular Value Decomposition(SVD)
- 90% of R is unknown, SVD is unsuitable
- Alernating Least Squares(ALS)

Movie Means Vs Time



Movie Means Vs Time



- Movie specific temporal plot.
- Sudden increase of mean at around 1500 days mark.
- The transient information is real and not artificially induced.
- The general trend is that the mean ratings tend to increase with time.

Movie : Darkwolf(2003)

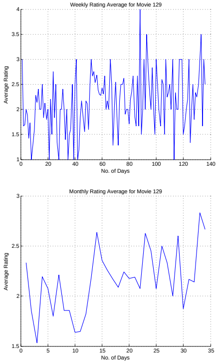


Figure: Weekly and Monthly Means for movies 129

Series : Six Feet Under: Season 4 (2004)

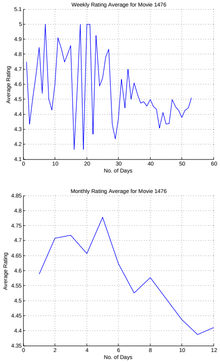


Figure: Weekly and Monthly Means for movie 1476

Singular Value Decomposition

Theorem (SVD)

For any matrix $A \in \mathcal{R}^{m \times n}$, *with* $m > n$, *there exists two orthogonal matrices* $U = (u_1, \ldots, u_m) \in \mathcal{R}^{m \times m}$ *&* $V = (v_1, \ldots, v_n) \in \mathcal{R}^{n \times n}$ *such that*

$$A = U \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^T,$$

where $\Sigma \in \mathcal{R}^{n \times n}$ *is diagonal matrix, i.e.,* $\Sigma = (\sigma_1, ..., \sigma_n)$, *with* $\sigma_1 \geq \sigma_2 \geq .... \geq \sigma_n \geq 0$. $\sigma_1, \ldots, \sigma_n$ *are called the singular values of* A. *Columns of* U *&* V *are called the right and left singular vectors of* A *respectively.*

Theorem (Eckart-Young theorem)

For a matrix $A \in \mathbb{R}^{m \times n}$, with $rank(A) = r > k$. The Frobenius norm of the matrix approximation problem

$$\min_{rank(\hat{A})=k} \|A - \hat{A}\| \tag{1}$$

has the solution of

$$\hat{A} = U_k \Sigma_k V_k^T, \tag{2}$$

where $U_k = \begin{pmatrix} u_1 & u_2 & \cdots & u_k \end{pmatrix}$, $V_k = \begin{pmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_n^T \end{pmatrix}$ *and*

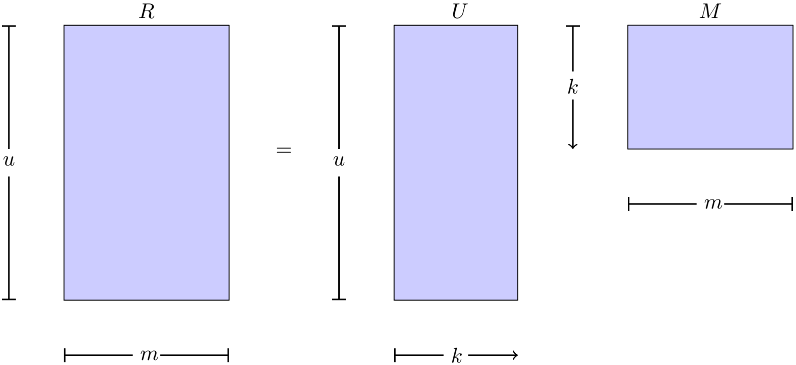
$\Sigma_k = diag(\sigma_1, \sigma_2, ..., \sigma_k)$.

Low Rank Matrix Approximation using SVD

$$A = \sum_{i=1}^{n} \sigma_i u_i v_i^T \approx \sum_{i=1}^{k} \sigma_i u_i v_i^T =: A_k \qquad k < r$$

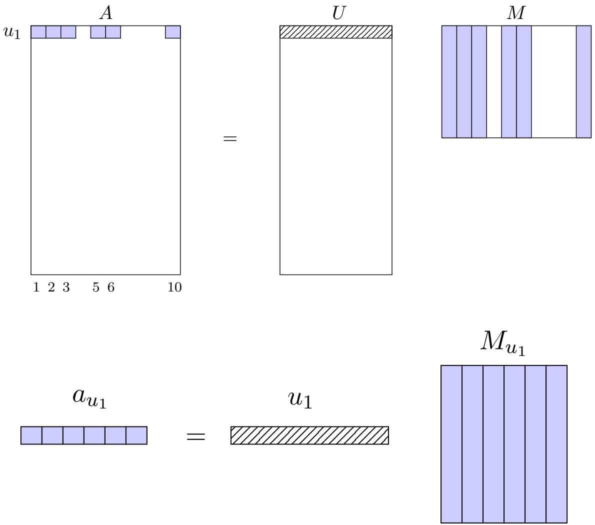
- The above approximation is based on the Eckart-Young theorem.
- It helps in removal of noise, solving ill-conditioned problems, and mainly in dimension reduction of data.

Alternating Least Squares



$$r_{ij} = \mathbf{u}_i \cdot \mathbf{m}_j \qquad \mathbf{u}_i \in \mathbb{R}^k \quad \mathbf{m}_j \in \mathbb{R}^k$$

- Models U,as *User feature space* and M,as *Movie feature space*.
- Similar to $Ax \approx b$, but with matrix RHS.
- Initialize M
  1. Solve $\min_U \|R - UM\|$, update U
  2. Solve $\min_M \|R - UM\|$, update M
- Repeat steps 1 and 2 until convergence


A
U
M
u_1
=
1 2 3 5 6 10
M_{u_1}
a_{u_1}
u_1
=

Static Bias

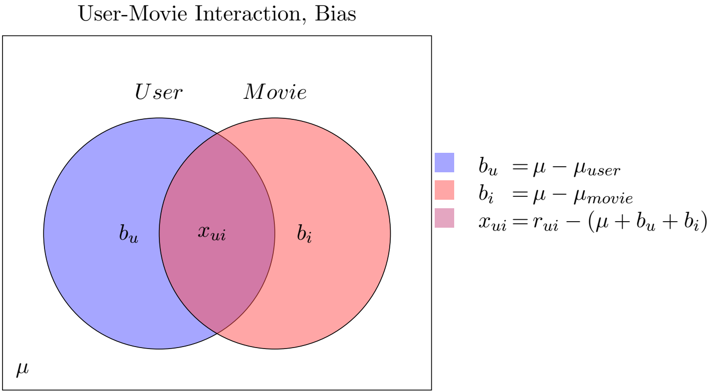


Figure: User-Movie Interaction

Temporal Bias

- We have captured the *Temporal bias* w.r.t movies only.
- Total period is of around 300 weeks. 30 bins of 10 week length.
- Each rating r_{ui} will fall under some bin.
- $b_{i,Bin(t)} = \mu_i - \mu_{Bin(t)}$
- *Baseline Predictor*, $b_{ui} = \mu + b_u + b_i + b_{i,Bin(t)}$

Initialization of M in ALS

We try different initialization techniques for M,

1. Initialize the M matrix, by assigning the means of movies as the first row entry, and small random numbers in the remaining entries.
2. $[U, S, V] = svd(R, 100)$
   $M = SV^T$

Prediction Models

1. Non-bias model
2. $NonTemp_{mu}$,
   $b_{ui} = \mu + b_u + b_i$
3. $Temp_{mu}$,
   $b_{ui} = \mu + b_u + b_i + b_{i,Bin(t)}$
4. $NonTemp_b$,
   $b_{ui} = b_u + b_i$
5. $Temp_b$,
   $b_{ui} = b_u + b_i + b_{i,Bin(t)}$

	SVD	ALS-I	ALS-II
1	Model 1	Model 6	Model 9
2	Model 3	-	Model 8
3	Model 2	-	Model 7
4	-	Model 4	-
5	-	Model 5	-

Table: Different Models

Evaluation Metric

RMSE: Root Means Square Error

$$RMSE = \left(\frac{\sum_{r_{i,j} \in \mathcal{T}} (\hat{r}_{ij} - r_{ij})^2}{|T|} \right)^{1/2}$$

RMSE: Non Biased SVD and ALS on Original data

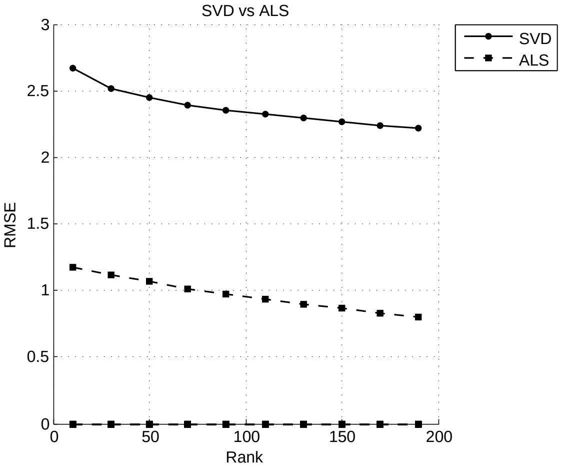


Figure: SVD - ALS

RMSE: Non Biased SVD

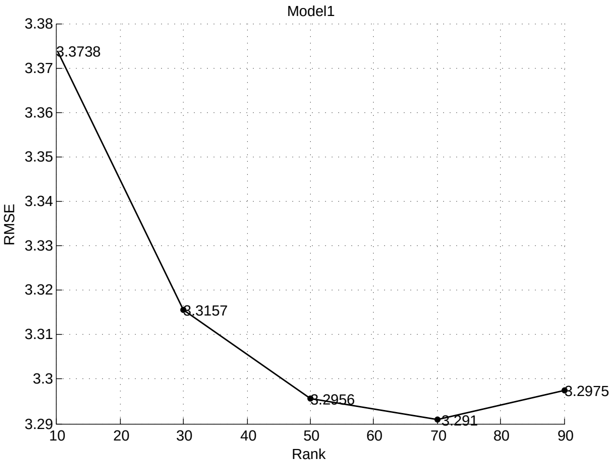


Figure: Model 1

RMSE: Static Biased SVD

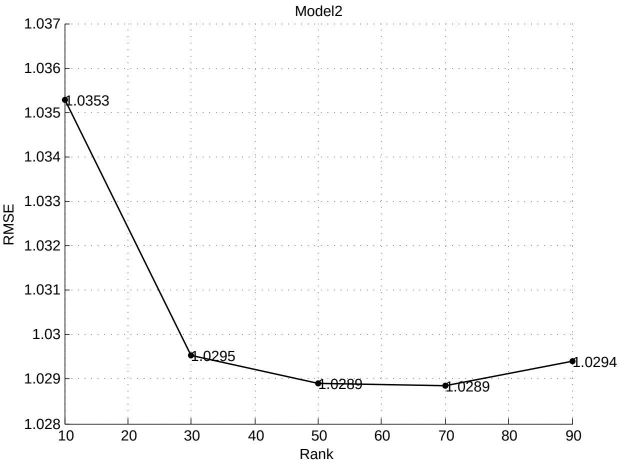


Figure: Model 2

RMSE: Temporal Biased SVD

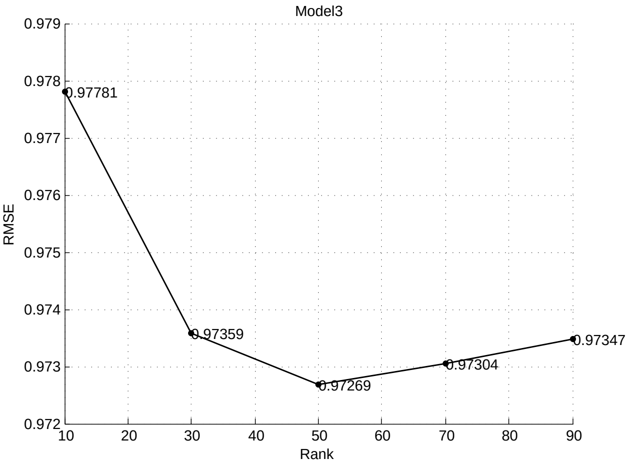
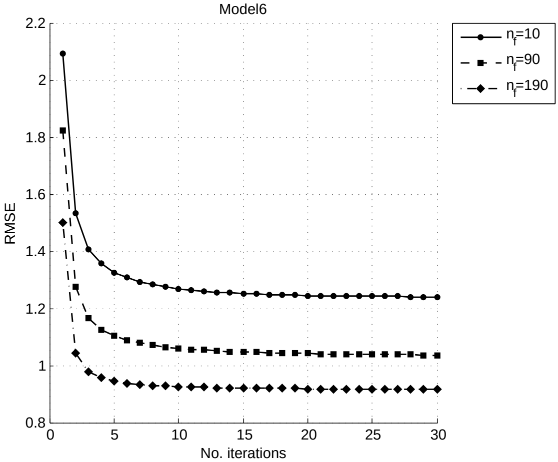
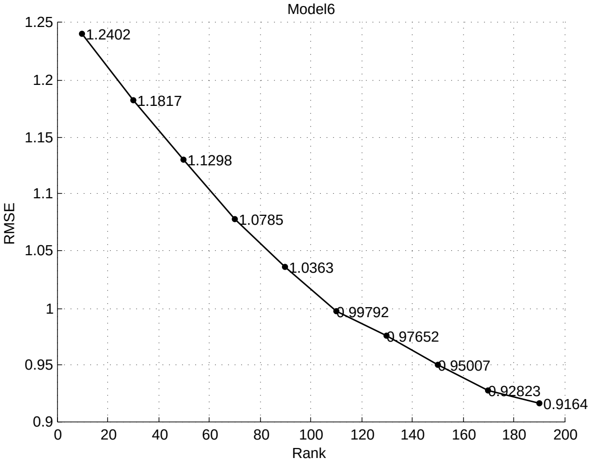


Figure: Model 3

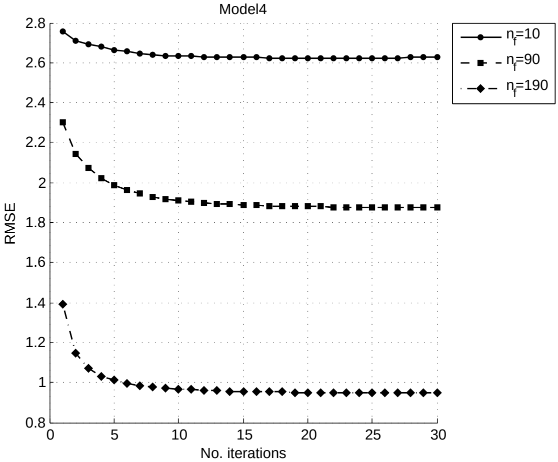
RMSE: Non Biased ALS-I



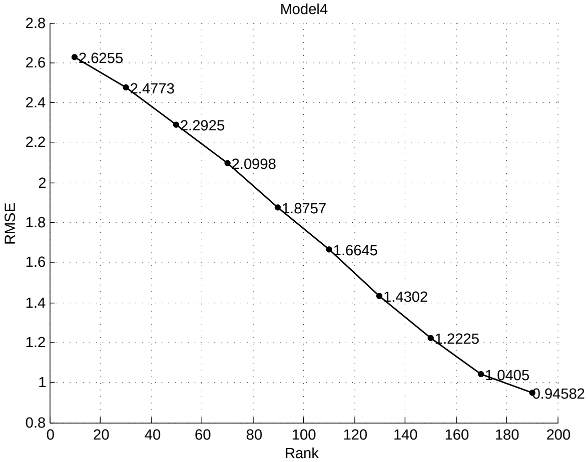
RMSE: Non Biased ALS-I



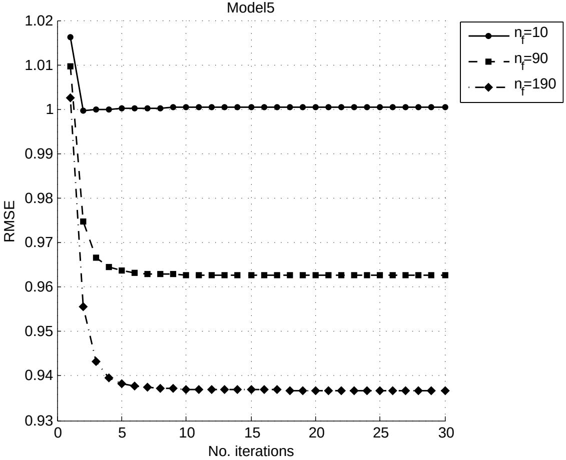
RMSE: Non Biased ALS-I



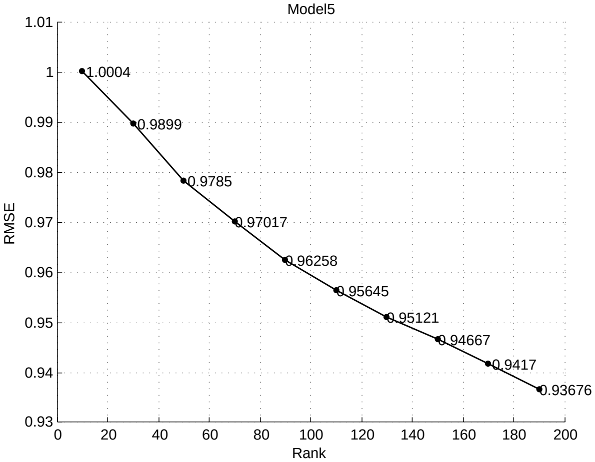
RMSE: Non Biased ALS-I



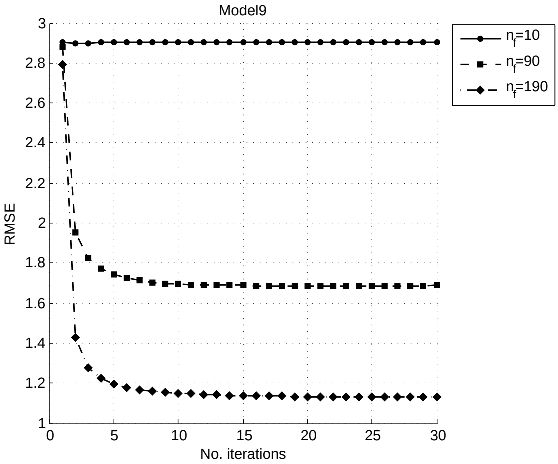
RMSE: Non Biased ALS-I



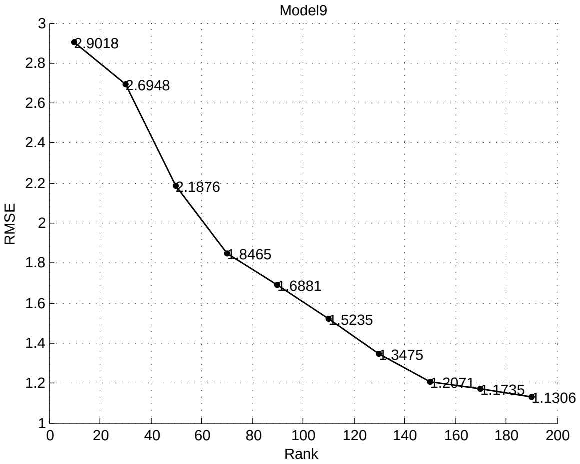
RMSE: Non Biased ALS-I



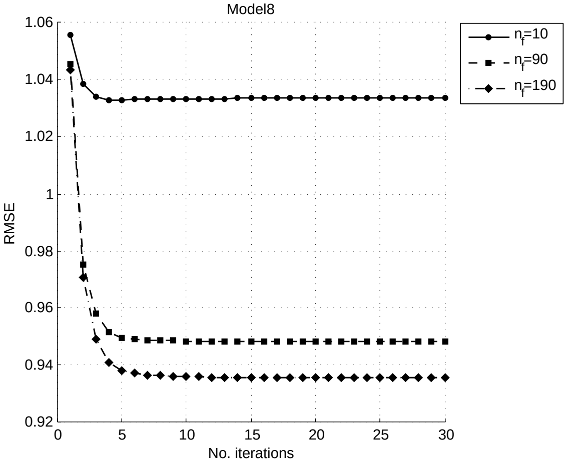
RMSE: Non Biased ALS-II



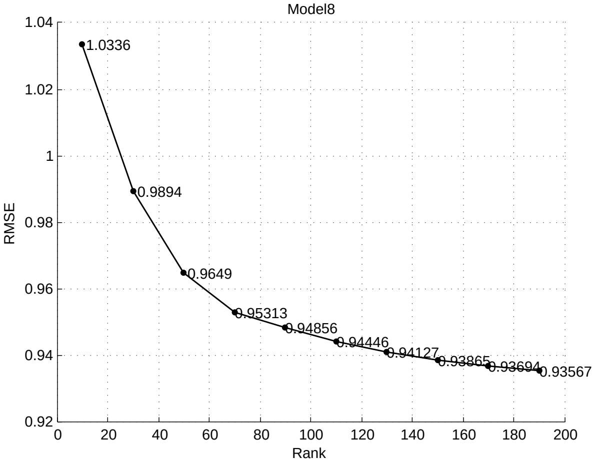
RMSE: Non Biased ALS-II



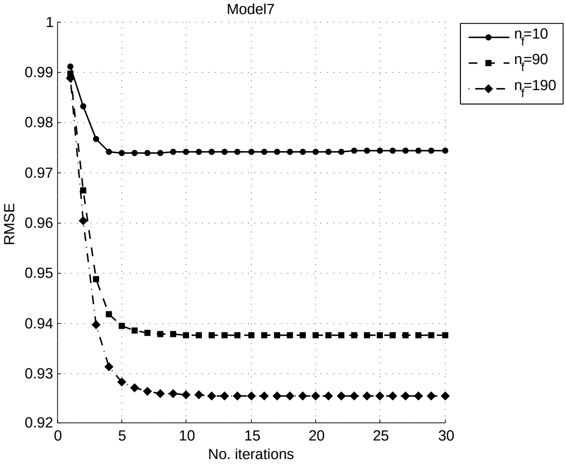
RMSE: Static Biased ALS-II



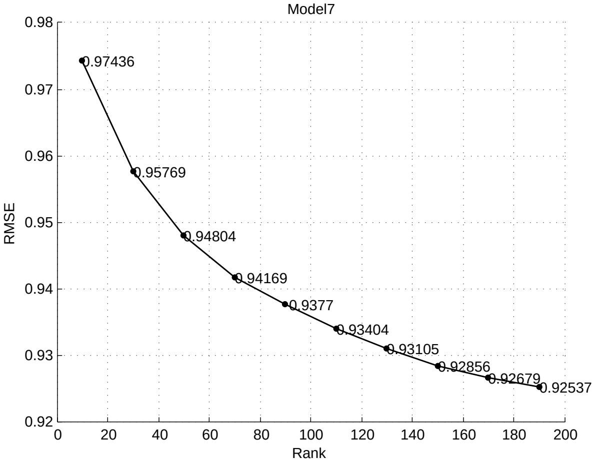
RMSE: Static Biased ALS-II



RMSE: Temporal Biased ALS-II



RMSE: Temporal Biased ALS-II



Conclusion

- Low-Rank approximation is useful for Collaborative Filtering.
- In case of SVD, best results are obtained for ranks ≈ 50.
- In case of ALS, there is no overfitting, best results for very high ranks.
- Less number of iterations required with high number of factors.
- Removal of biases helps in collaborative filtering, since it is the interaction which we are trying to study and factorize.

The End