SVD++ and Probabilistic Matrix Factorization

Claudio Rojas Bórquez

References

This presentation is a summary of the following papers:

- Yehuda, Koren. *Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model.* Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. Pages 426-434 [2008]

- Salakhutdinov, Ruslan. *Probabilistic Matrix Factorization.* [2007]

In this presentation

- Collaborative filtering models
- SVD++
- Probabilistic matrix approximation

Collaborative Filtering

- CF is a methodology where past transactions are analysed in order to establish connections between users and products.

- The two more successful approaches to CF are latent factor models, represented by SVD, and neighbourhood models (Yehuda, 2008)

- Both models have strengths and weaknesses.

Latent Factor models

- What it does?

Transform items and users to the same latent factor space, thus making them directly comparable. The most common way to do so is SVD

$$M = U\Sigma V^T \rightarrow \tilde{M} = U\tilde{\Sigma}V^T$$

The approximating matrix is the best approximation in the Frobenius norm (more on this later)

Latent Factor models

- The drawback of latent factor models are basically two:
  - They show poor performance at detecting strong associations among a small set of closely related items.
  - Their results is little to no explanation on the reasons behind a recommendation which is bad for the user who *wants to know why an item is being recommended to him/her.*
- SVD also is sensible to sparse data which is very common in the field.

Neighborhood models

- Neighborhood method takes in account either items or users that are similar between themselves, this is usually made using a *similarity score* s_{ij}

- This method is very good at detecting localised relationships which is exactly where Latent method behave poorly.

- They are blind to the *big structure.*

Implicit feedback

- Neither of both previous models include implicit feedback, this in one of the goals the author wants to achieve with the SVD++ model.
- There are many kinds of implicit feedback, some are not so obvious, for example in the case of the Netflix's data that a user rates a movie, without regard of the actual rating, may be considered implicit feedback.

SVD++ model

- This model tries to mix strengths of the latent model as well of the neighbourhood model.

- The prediction made by this model has 3 parts

$$\begin{aligned}\hat{r}_{ui} = \quad & \mu + b_u + b_i \\ & + q_i^T \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right) \\ & + |R^k(i;u)|^{-\frac{1}{2}} \sum_{j \in R^k(i;u)} (r_{uj} - b_{uj}) w_{ij} + |N^k(i;u)|^{-\frac{1}{2}} \sum_{j \in N^k(i;u)} c_{ij}\end{aligned}$$

1st tier

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right) + |R^k(i;u)|^{-\frac{1}{2}} \sum_{j \in R^k(i;u)} (r_{uj} - b_{uj}) w_{ij} + |N^k(i;u)|^{-\frac{1}{2}} \sum_{j \in N^k(i;u)} c_{ij}$$

The first term is the basis rate, it takes in account a global mean and the bias of both user and item.

2nd tier

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right) + |R^k(i;u)|^{-\frac{1}{2}} \sum_{j \in R^k(i;u)} (r_{uj} - b_{uj}) w_{ij} + |N^k(i;u)|^{-\frac{1}{2}} \sum_{j \in N^k(i;u)} c_{ij}$$

The second term is similar to the original SVD but takes in account the implicit feedback present in the set of rated items N(u)

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right) + |R^k(i;u)|^{-\frac{1}{2}} \sum_{j \in R^k(i;u)} (r_{uj} - b_{uj}) w_{ij} + |N^k(i;u)|^{-\frac{1}{2}} \sum_{j \in N^k(i;u)} c_{ij}$$

The third and fourth terms are the neighborhood terms. The former is the weighted bias of the basis rate and the actual rate and the latter is the local effect of the implicit feedback

Actual formulation

- If $e_{ui} = r_{ui} - \hat{r}_{ui}$ then the problem that is solved is:

$$\min \sum_{u,i} e_{ui}^2 + \lambda \|\text{params}\|^2$$

- This optimisation can be performed using gradient descent.

SVD++ performance

Model	50 factors	100 factors	200 factors
SVD	0.9046	0.9025	0.9009
Asymmetric-SVD	0.9037	0.9013	0.9000
SVD++	0.8952	0.8924	0.8911

Table 1: Comparison of SVD-based models: prediction accuracy is measured by RMSE on the Netflix test set for varying number of factors (f). Asymmetric-SVD offers practical advantages over the known SVD model, while slightly improving accuracy. Best accuracy is achieved by SVD++, which directly incorporates implicit feedback into the SVD model.

SVD++ performance

	50 factors	100 factors	200 factors
RMSE	0.8877	0.8870	0.8868
time/iteration	17min	20min	25min

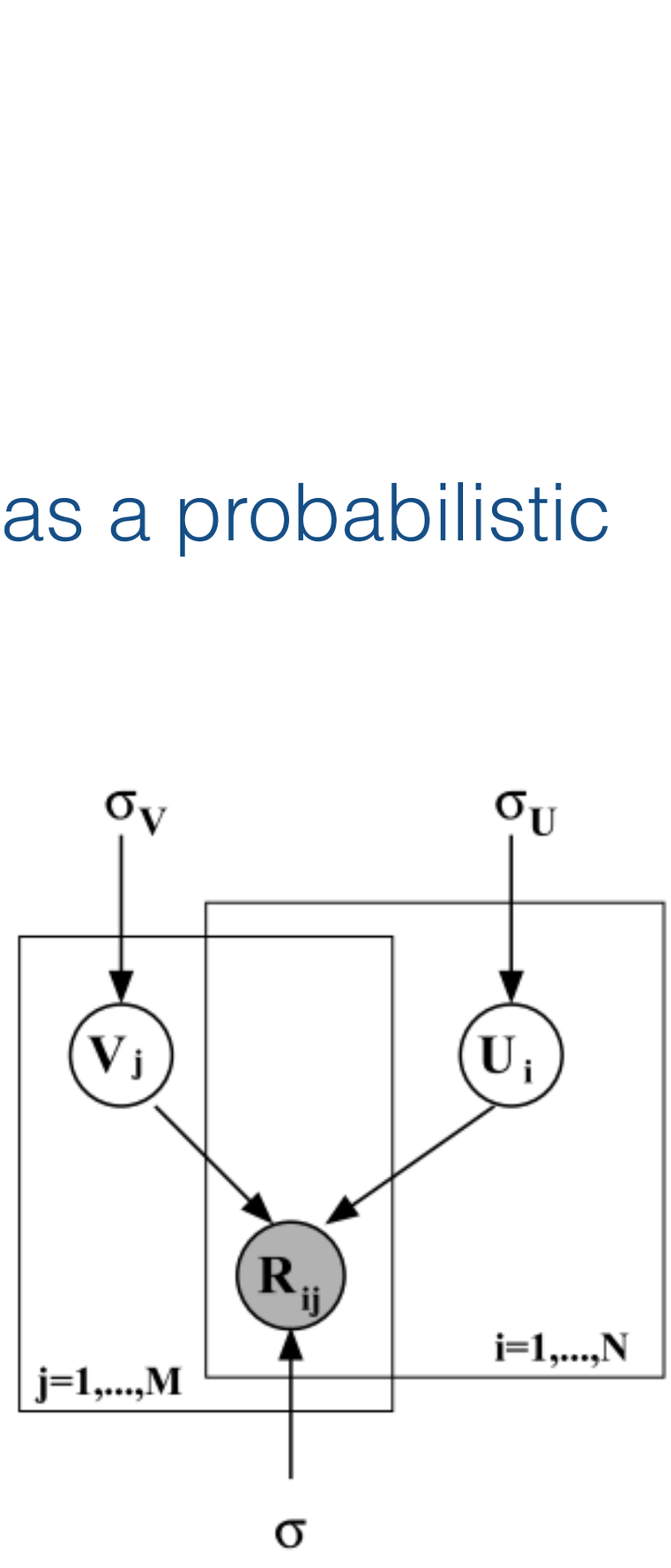
Table 2: Performance of the integrated model. Prediction accuracy is improved by combining the complementing neighborhood and latent factor models. Increasing the number of factors contributes to accuracy, but also adds to running time.

PMF (Probabilistic matrix factorization)

- This method tries to make a matrix decomposition as SVD but it has two main differences:
  - It only takes in account the non zero elements, i.e performs well with sparse data
  - Scales linearly with the number of observations

PMF

- This model view the rating as a probabilistic graphical model:



$$p(R|U,V,\sigma^2) = \prod_i \prod_j \left[\mathcal{N}(R_{ij}|U_i^T V_j, \sigma^2)\right]_{ij}^I$$

$$p(U|\sigma_u^2) = \prod_i \mathcal{N}(U_i|0,\sigma_u^2 I) \qquad p(V|\sigma_v^2) = \prod_j \mathcal{N}(V_j|0,\sigma_v^2 I)$$

$$\begin{aligned}\ln p(U,V|R,\sigma^2,\sigma_V^2,\sigma_U^2) = &-\frac{1}{2\sigma^2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij}-U_i^T V_j)^2 - \frac{1}{2\sigma_U^2}\sum_{i=1}^{N} U_i^T U_i - \frac{1}{2\sigma_V^2}\sum_{j=1}^{M} V_j^T V_j \\ &-\frac{1}{2}\left(\left(\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\right)\ln\sigma^2 + ND\ln\sigma_U^2 + MD\ln\sigma_V^2\right) + C, \qquad (3)\end{aligned}$$

PMF

- Given prior for the latent factors for users and items the equivalent problem is minimise the square error given by:

$$E = \frac{1}{2}\sum_i \sum_j I_{ij}(R_{ij} - U_i^T V_j) + \frac{\lambda_u}{2}\sum_i \|U_i\|^2 + \frac{\lambda_v}{2}\|V_j\|^2$$

- It's posible to put priors to the parameters which receives the name of adaptative PMF

PMF performance

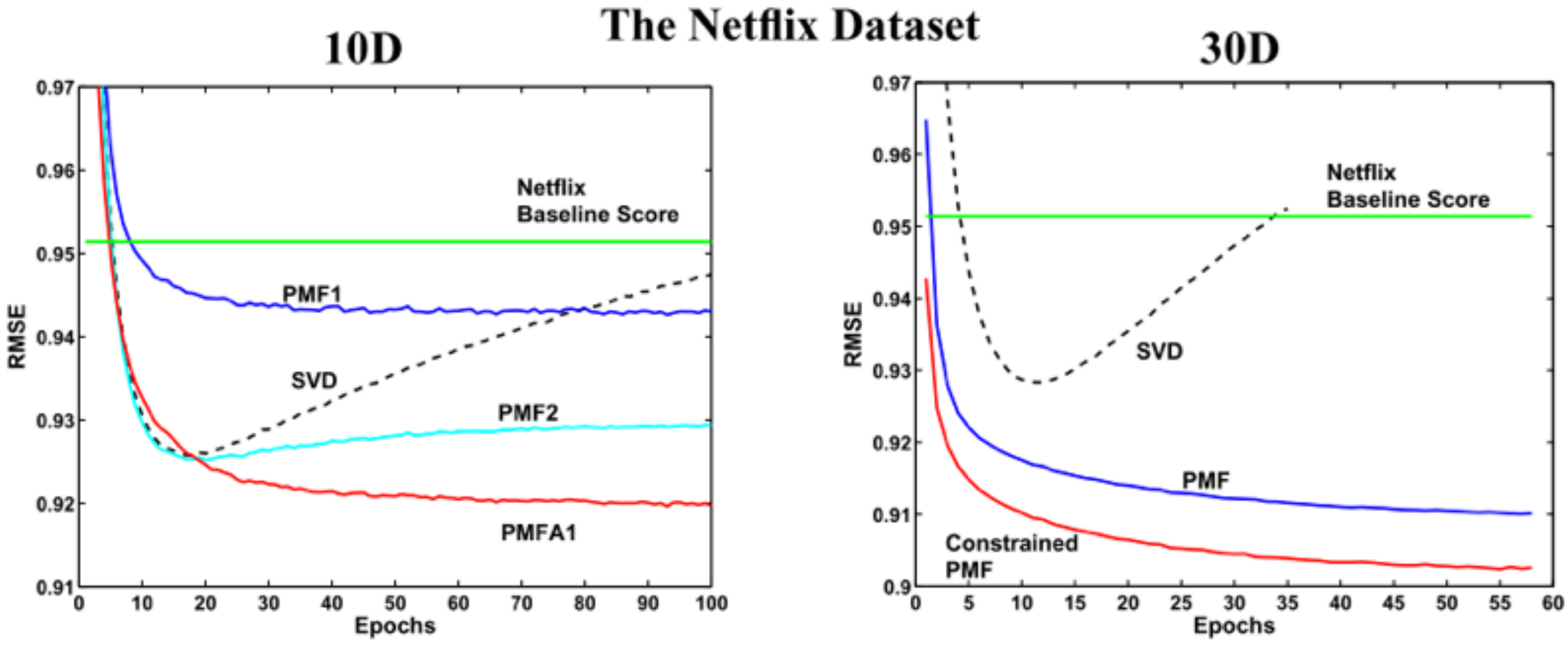
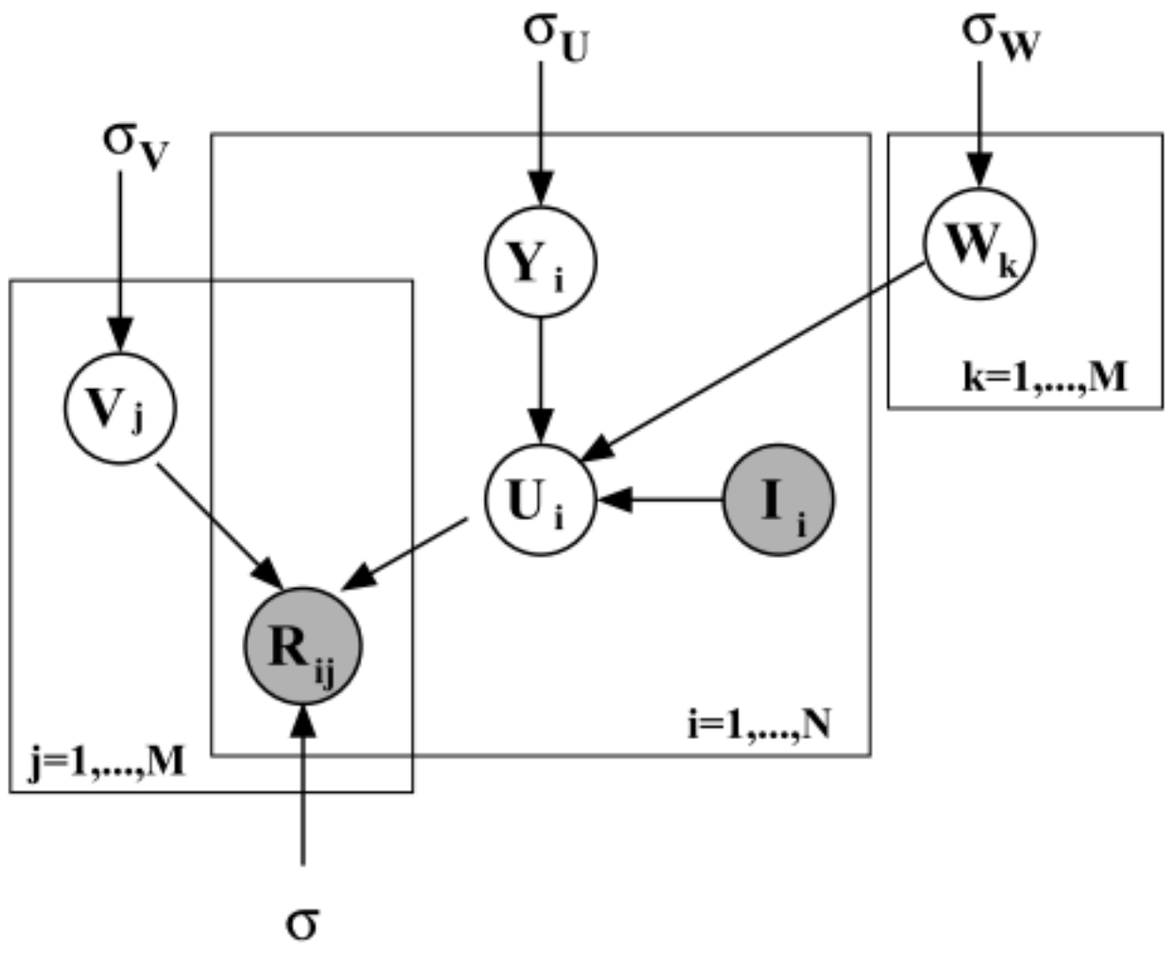


Figure 2: Left panel: Performance of SVD, PMF and PMF with adaptive priors, using 10D feature vectors, on the full Netflix validation data. Right panel: Performance of SVD, Probabilistic Matrix Factorization (PMF) and constrained PMF, using 30D feature vectors, on the validation data. The y-axis displays RMSE (root mean squared error), and the x-axis shows the number of epochs, or passes, through the entire training dataset.

Constrained PMF



Constrained PMF

$$U_i = Y_i + \frac{\sum_{k=1}^{M} I_{ik} W_k}{\sum_{k=1}^{M} I_{ik}}.$$

$$p(W|\sigma_W) = \prod_{k=1}^{M} \mathcal{N}(W_k|0, \sigma_W^2 \mathbf{I}).$$

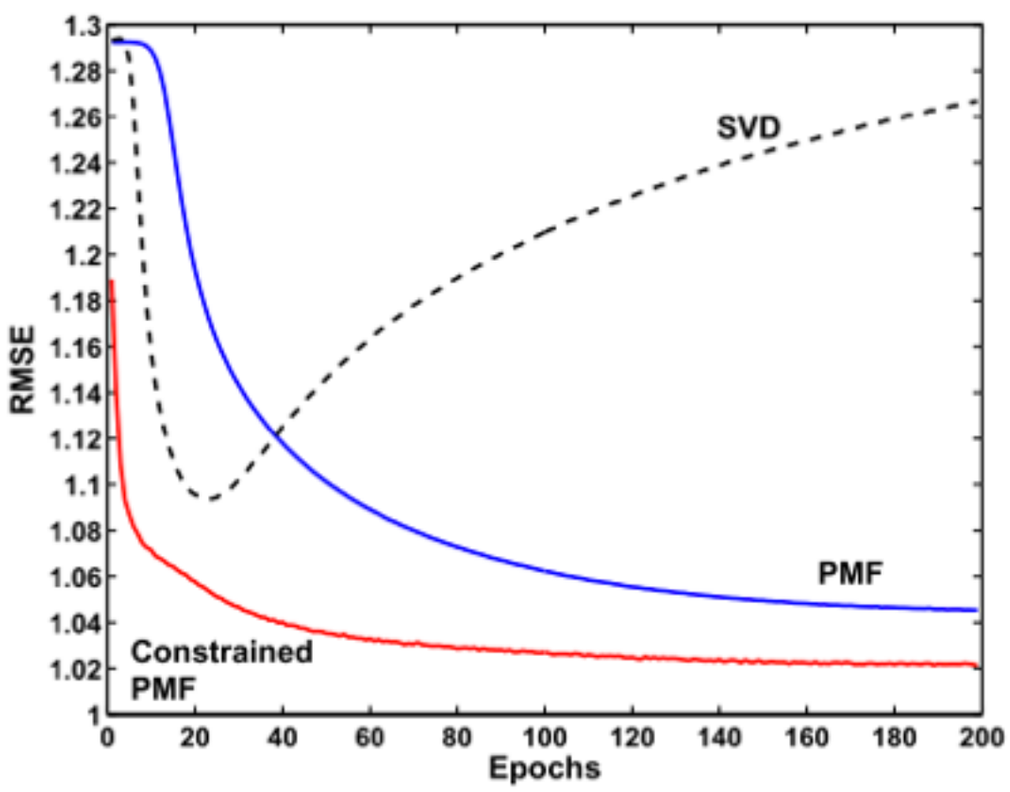
$$p(R|Y, V, W, \sigma^2) = \prod_{i=1}^{N} \prod_{j=1}^{M} \left[\mathcal{N}(R_{ij}|g([Y_i + \frac{\sum_{k=1}^{M} I_{ik} W_k}{\sum_{k=1}^{M} I_{ik}}]^T V_j), \sigma^2) \right]^{I_{ij}}.$$

Constrained PMF

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\left(R_{ij} - g\left(\left[Y_i + \frac{\sum_{k=1}^{M} I_{ik}W_k}{\sum_{k=1}^{M} I_{ik}}\right]^T V_j\right)\right)^2$$

$$+\frac{\lambda_Y}{2}\sum_{i=1}^{N} \parallel Y_i \parallel_{Fro}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M} \parallel V_j \parallel_{Fro}^2 + \frac{\lambda_W}{2}\sum_{k=1}^{M} \parallel W_k \parallel_{Fro}^2,$$

Constrained PMF



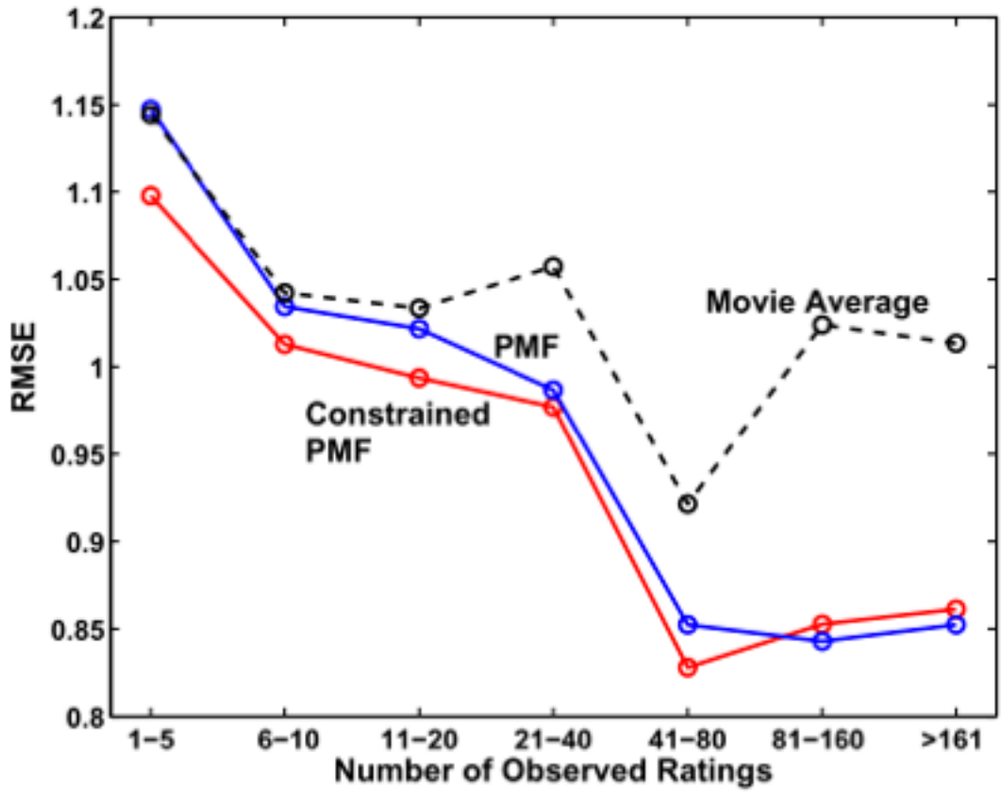


Figure 3: Left panel: Performance of SVD, Probabilistic Matrix Factorization (PMF) and constrained PMF on the validation data. The y-axis displays RMSE (root mean squared error), and the x-axis shows the number of epochs, or passes, through the entire training dataset. Right panel: Performance of constrained PMF, PMF, and the movie average algorithm that always predicts the average rating of each movie. The users were grouped by the number of observed ratings in the training data.