Matrix factorization methods for collaborative filtering

Andriy Mnih and Ruslan Salakhutdinov
University of Toronto, Machine Learning Group

What is collaborative filtering?

- The goal of collaborative filtering (CF) is to infer user preferences for items given a large but incomplete collection of preferences for many users.
- For example:
  - Suppose you infer from the data that most of the users who like “Star Wars” also like “Lord of the Rings” and dislike “Dune”.
  - Then if a user watched and liked “Star Wars” you would recommend him/her “Lord of the Rings” but not “Dune”.

- Peferences can be explicit or implicit:
  - Explicit preferences: ratings given to items by users.
  - Implicit preferences: which items were rented or bought by users.

Collaborative filtering vs. content-based filtering

- Content-based filtering makes recommendations based on item content.
  - E.g. for a movie: genre, actors, director, length, language, number of car chases, etc.
  - Can be used to recommend new items for which no ratings are available yet.
  - Does not perform as well as collaborative filtering in most cases.

- Collaborative filtering does not look at item content.
  - Preferences are inferred from rating patterns alone.
  - Cannot recommend new items – they all look the same to the system.
  - Very effective when a sufficient amount of data is available.

Netflix Prize: In it for the money

- Two years ago, Netflix has announced a movie rating predictions competition.

- Whoever improves Netflix's own baseline score by 10% will win the 1 million dollar prize.

- The training data set consists of 100,480,507 ratings from 480,189 randomly-chosen, anonymous users on 17,770 movie titles. The data is very sparse, most users rate only few movies.

- Also, Netflix provides a test set containing 2,817,131 user/movie pairs with the ratings withheld. The goal is to predict those ratings as accurately as possible.

Course projects

- We will provide you with a subset of the Netflix training data: a few thousand users + a few thousand movies, so that you can easily run your algorithms on CDF machines.

- We will also provide you with a validation set. You will report the achieved prediction accuracy on this validation set.

- There will be two projects based on the following two models:
  - Probabilistic Matrix Factorization (PMF)
  - Restricted Boltzmann Machines (RBM's)
- You can choose which model you would like to work on.
- This tutorial will cover only PMF (the easy 4-5% on Netflix).

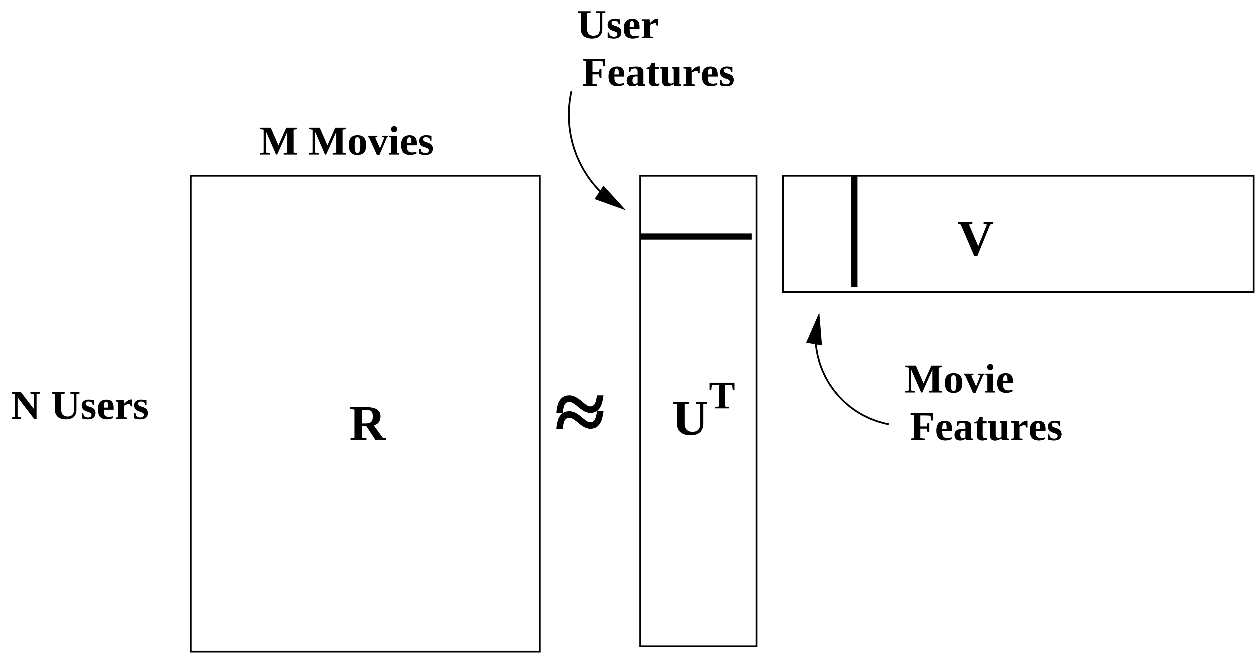
CF as matrix completion

- Collaborative filtering can be viewed as a matrix completion problem.

	Alice	Bob	Alan	Claude	Norbert
The Seventh Seal	★★★★★	★★	★	★★★★★	★★★★
Metropolis		★	★	★★★★	★★★★★
Akira	★★★★	★★★★	★★★		★★★★★
Miss Congeniality	★★	★★★	★★★★	★★	★
Independence Day	★	★★★★★	★★★★★	★★	★
Pulp Fiction	★★★★★	★★★★★		★★★	

- Task: given a user/item matrix with only a small subset of entries present, fill in (some of) the missing entries.
- Perhaps the simplest effective way to do this is to factorize the rating matrix into a product of two smaller matrices.

Matrix factorization: notation



- Suppose we have M movies, N users, and integer rating values from 1 to K.

- Let R_{ij} be the rating of user i for movie j, and $U \in R^{D\times N}$, $V \in R^{D\times M}$ be latent user and movie feature matrices.

- We will use U_i and V_j to denote the latent feature vectors for user i and movie j respectively.

Matrix factorization: the non-probabilistic view

- To predict the rating given by user i to movie j, we simply compute the dot product between the corresponding feature vectors:
  - $\hat{R}_{ij} = U_i^T V_j = \sum_k U_{ik} V_{jk}$
- Intuition: for each user, we predict a movie rating by giving the movie feature vector to a linear model.
  - The movie feature vector can be viewed as the input.
  - The user feature vector can be viewed as the weight vector.
  - The predicted rating is the output.
  - Unlike in linear regression, where inputs are fixed and weights are learned, we learn *both* the weights *and* the inputs (by minimizing squared error).
  - Note that the model is symmetric in movies and users.

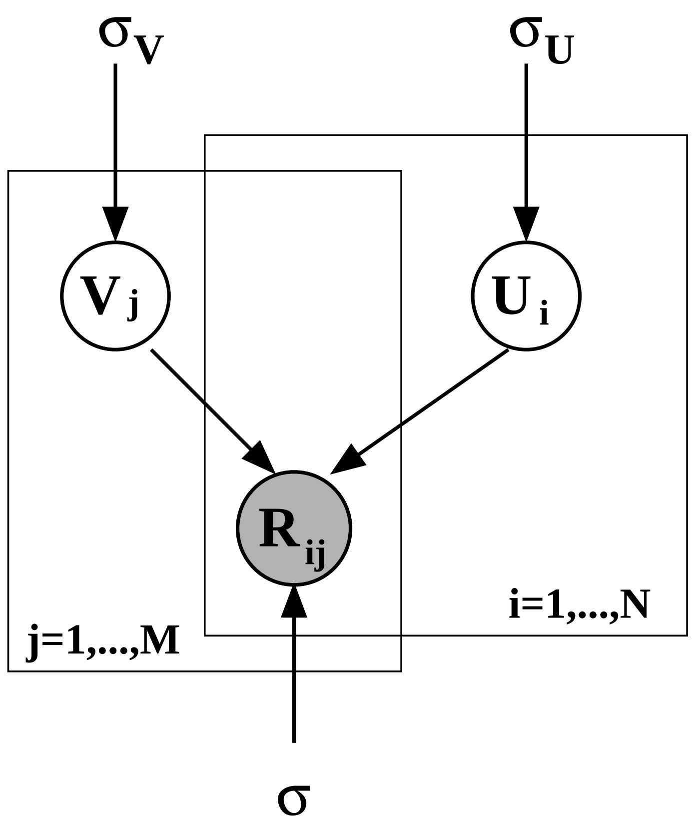
Probabilistic Matrix Factorization (PMF)

- PMF is a simple probabilistic linear model with Gaussian observation noise.

- Given the feature vectors for the user and the movie, the distribution of the corresponding rating is:

$$p(R_{ij}|U_i, V_j, \sigma^2) = \mathcal{N}(R_{ij}|U_i^T V_j, \sigma^2)$$

- The user and movie feature vectors are given zero-mean spherical Gaussian priors:

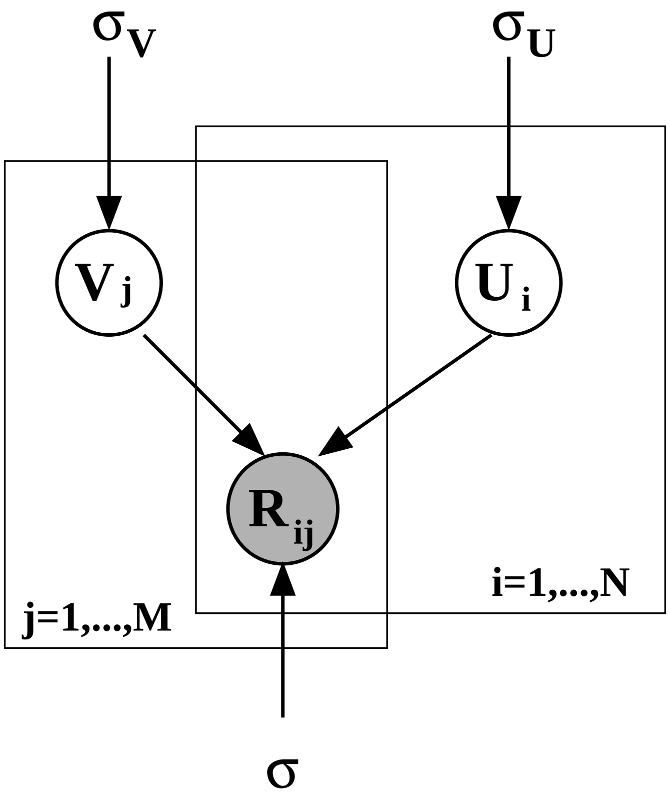


$$p(U|\sigma_U^2) = \prod_{i=1}^{N} \mathcal{N}(U_i|0, \sigma_U^2 \mathrm{I}), \quad p(V|\sigma_V^2) = \prod_{j=1}^{M} \mathcal{N}(V_j|0, \sigma_V^2 \mathrm{I})$$

Learning (I)

- MAP Learning: Maximize the log-posterior over movie and user features with fixed hyperparameters.

- Equivalent to minimizing the sum-of-squared-errors with quadratic regularization terms:



$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\left(R_{ij} - U_i^T V_j\right)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N} \| U_i \|_{Fro}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M} \| V_j \|_{Fro}^2$$

$\lambda_U = \sigma^2/\sigma_U^2$, $\lambda_V = \sigma^2/\sigma_V^2$, and $I_{ij} = 1$ if user i rated movie j and is 0 otherwise.

Learning (II)

$$\boldsymbol{E} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} \boldsymbol{I}_{ij}\left(\boldsymbol{R}_{ij} - \boldsymbol{U}_i^T \boldsymbol{V}_j\right)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N} \parallel \boldsymbol{U}_i \parallel_{Fro}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M} \parallel \boldsymbol{V}_j \parallel_{Fro}^2$$

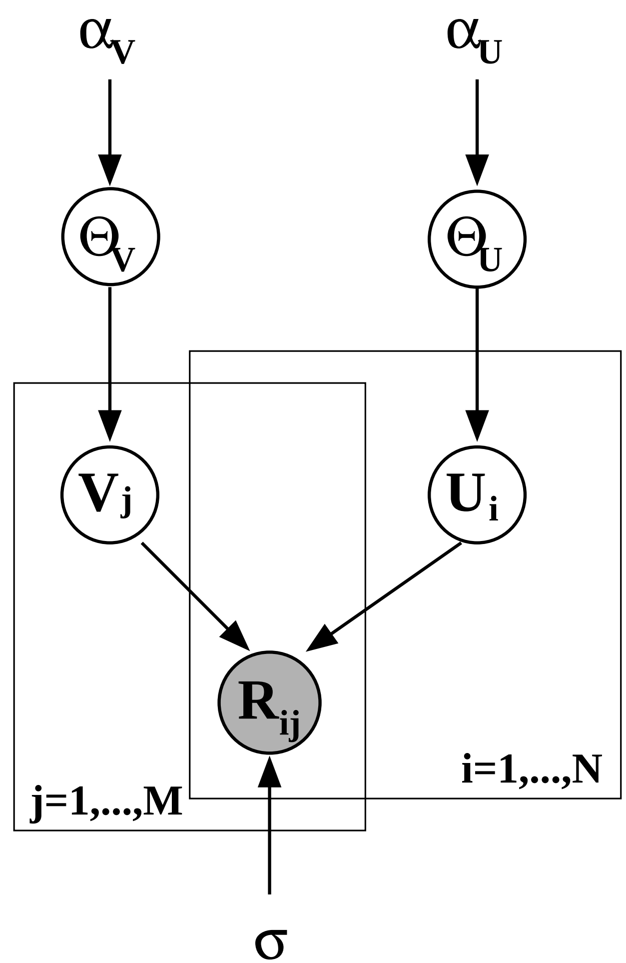
- Find a local minimum by performing gradient descent in $\boldsymbol{U}$ and $\boldsymbol{V}$.
- If all ratings were observed, the objective reduces to the SVD objective in the limit of prior variances going to infinity.
- PMF can be viewed as a probabilistic extension of SVD, which works well even when most entries in $\boldsymbol{R}$ are missing.

Automatic Complexity Control for PMF (I)

- Model complexity is controlled by the noise variance σ^2 and the parameters of the priors (σ_U^2 and σ_V^2).

- Approach: Find a MAP estimate for the hyperparameters after introducing priors for them.

- Learning: Find a point estimate of parameters and hyperparameters by maximizing the log-posterior:

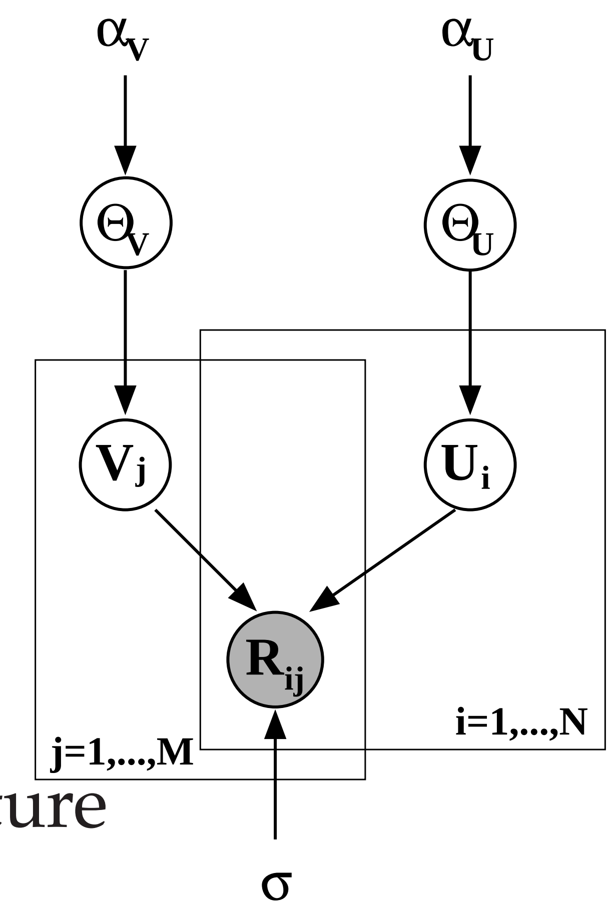


$$\ln p(U, V, \sigma^2, \Theta_U, \Theta_V | R) = \ln p(R|U, V, \sigma^2) + \ln p(U|\Theta_U) + \ln p(V|\Theta_V) + \ln p(\Theta_U) + \ln p(\Theta_V) + C$$

Automatic Complexity Control for PMF (II)

- Can use more sophisticated regularization methods than simple penalization of the Frobenius norm of the feature matrices:

  - priors with diagonal or full covariance matrices and adjustable means, or even mixture of Gaussians priors.



- Using spherical Gaussian priors for feature vectors leads to the standard PMF with λ_U and λ_V chosen automatically.

- Automatic selection of the hyperparameter values worked considerably better than the manual approach that used a validation set.

Constrained PMF (I)

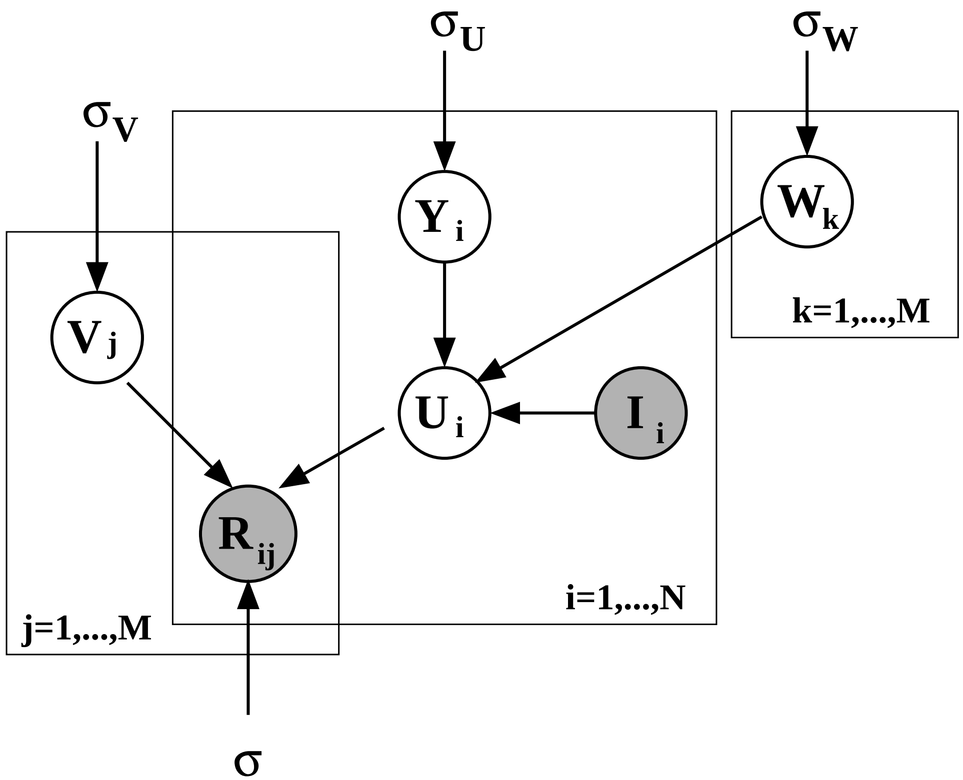
- Two users that have rated similar movies are likely to have preferences more similar than two randomly chosed users.
- Make the prior for the user feature vector depend on the movies the user has rated.
- This will force users who have seen the same (or similar) movies to have similar prior distributions for their feature vectors.

Constrained PMF (II)

- Let $\boldsymbol{W} \in \boldsymbol{R}^{D \times M}$ be a latent similarity constraint matrix.

- We define the feature vector for user i as:

$$\boldsymbol{U}_i = \boldsymbol{Y}_i + \frac{\sum_{k=1}^{M} \boldsymbol{I}_{ik} \boldsymbol{W}_k}{\sum_{k=1}^{M} \boldsymbol{I}_{ik}}$$



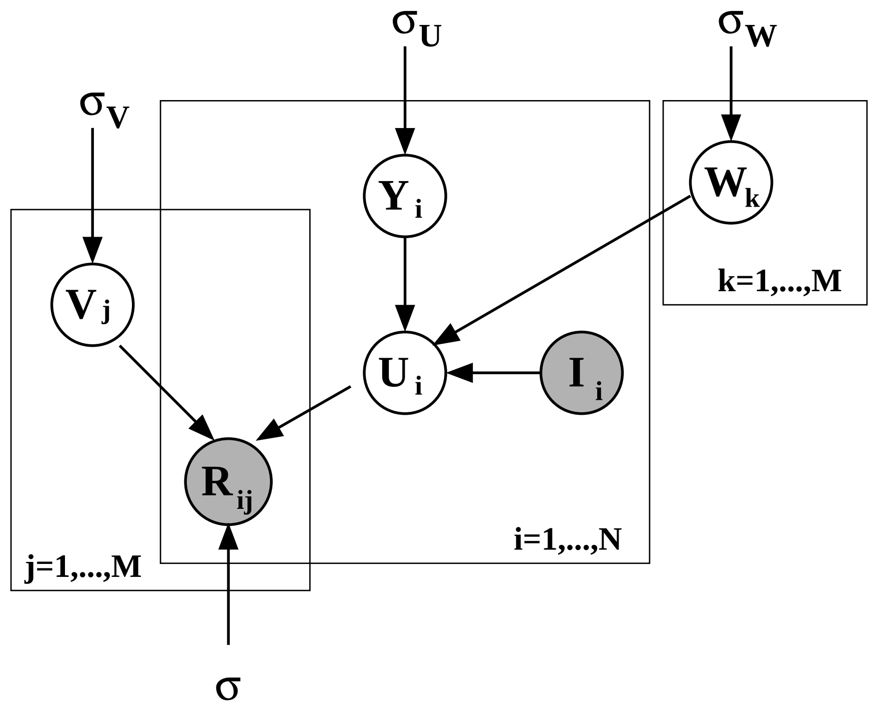
- $\boldsymbol{I}$ is the observed indicator matrix, $\boldsymbol{I}_{ij} = 1$ if user i rated movie j and 0 otherwise.

- Performs considerably better on infrequent users.

Constrained PMF (III)

- The feature vector for user i:

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



- For standard PMF, U_i and Y_i are equal because the prior mean is fixed at zero.

- The model:

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

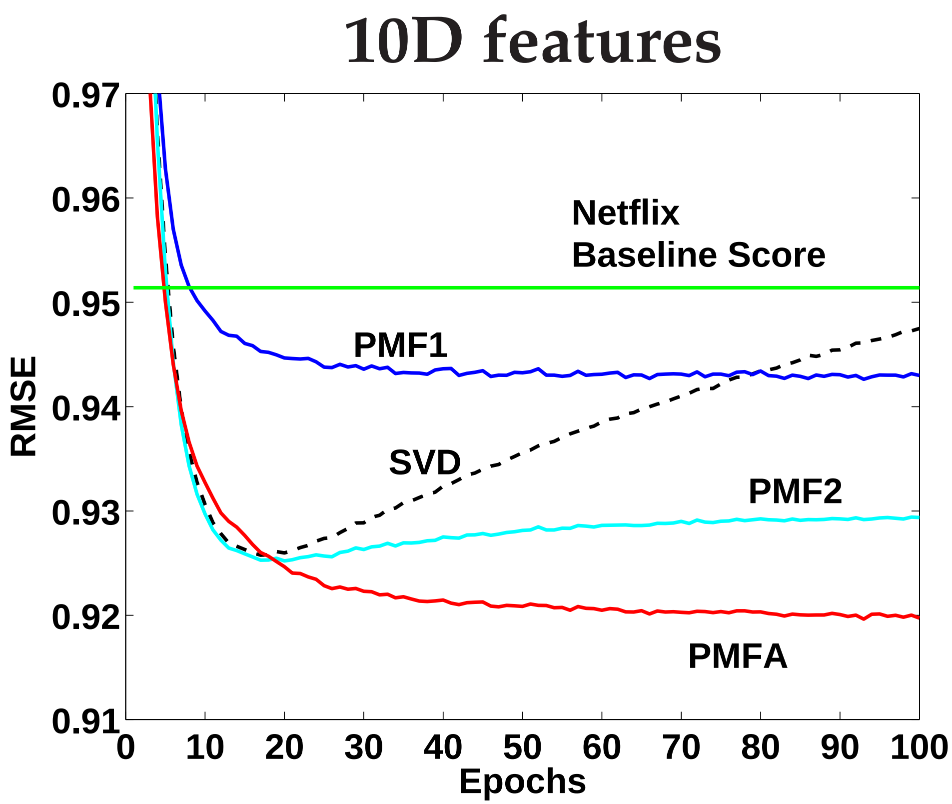
with

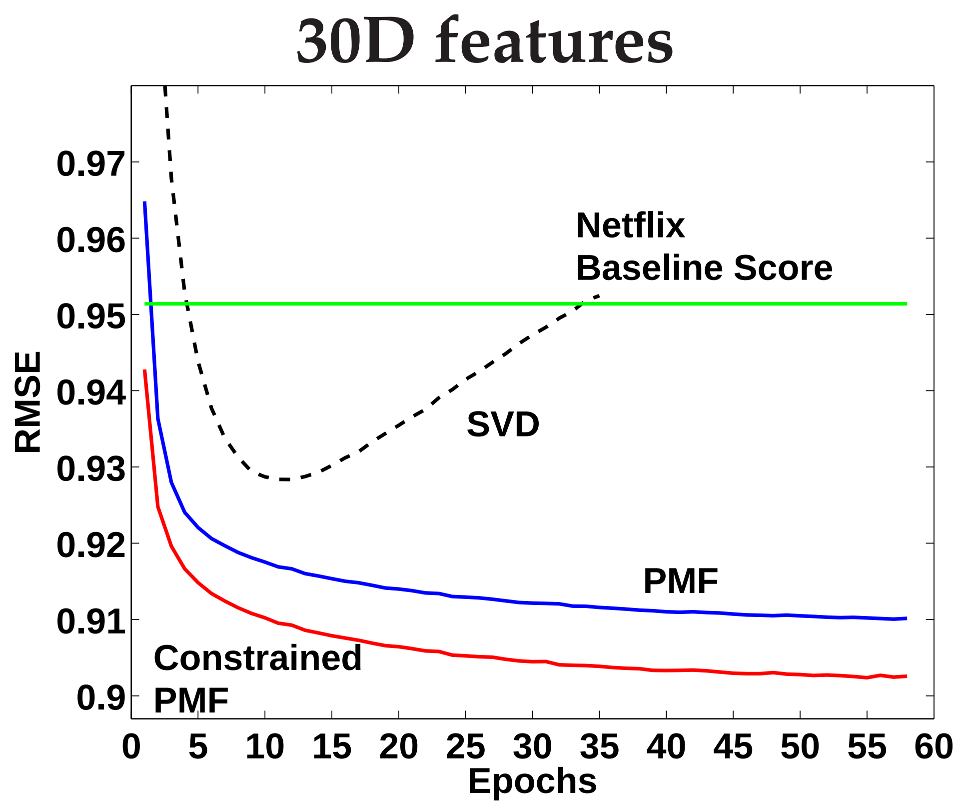
$$p(W|\sigma_W) = \prod_{k=1}^{M} \mathcal{N}(W_k|0,\sigma_W^2 \mathrm{I})$$

The Netflix Dataset

- The Netflix dataset is large, sparse, and imbalanced.
- The training set: 100,480,507 ratings from 480,189 users on 17,770 movies.
- The validation set: 1,408,395 ratings. The test set: 2,817,131 user/movie pairs with ratings withheld.
- The dataset is very imbalanced. The number of ratings entered by each user ranges from 1 to over 15000.
- Performance is assessed by submitting predictions to Netflix, which prevents accidental cheating since the test answers are known only to Netflix.

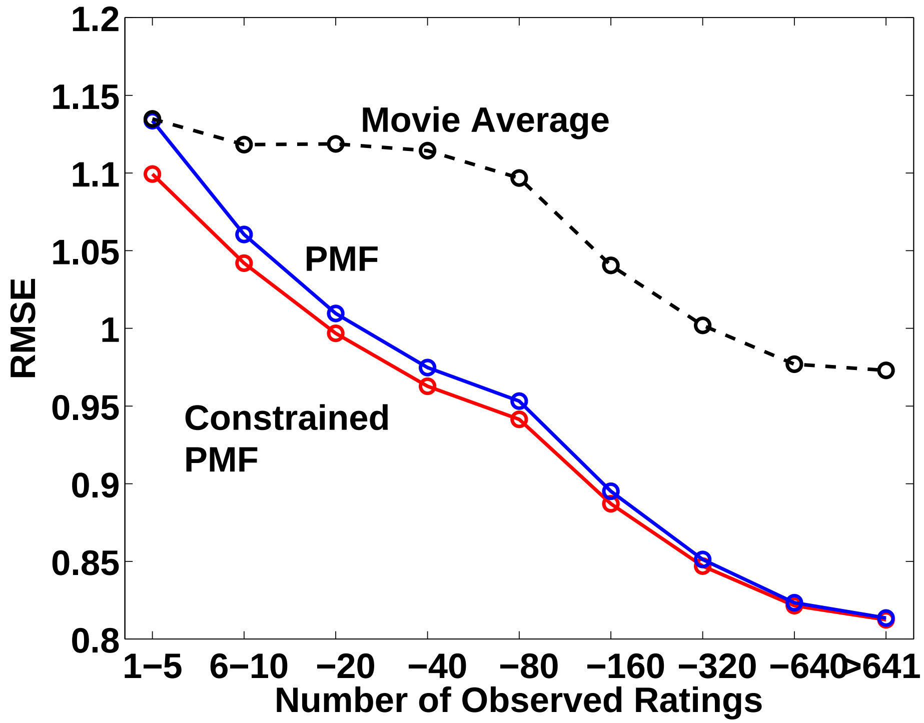
Experimental Results

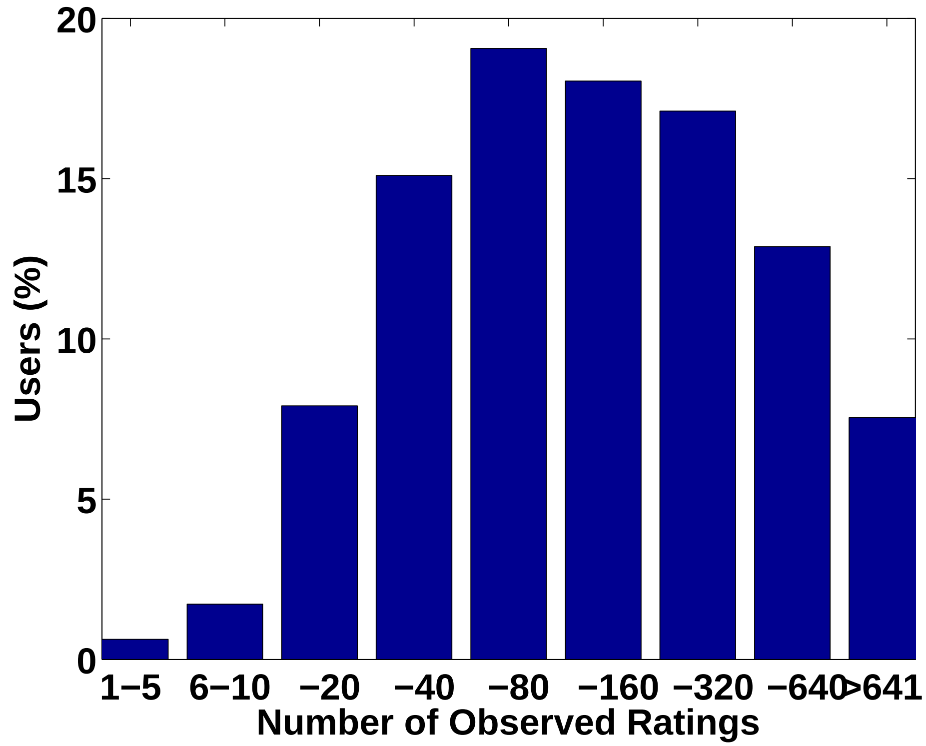




- Performance of SVD, PMF and PMF with adaptive priors, using 10D and 30D feature vectors, on the full Netflix validation set.

Experimental Results

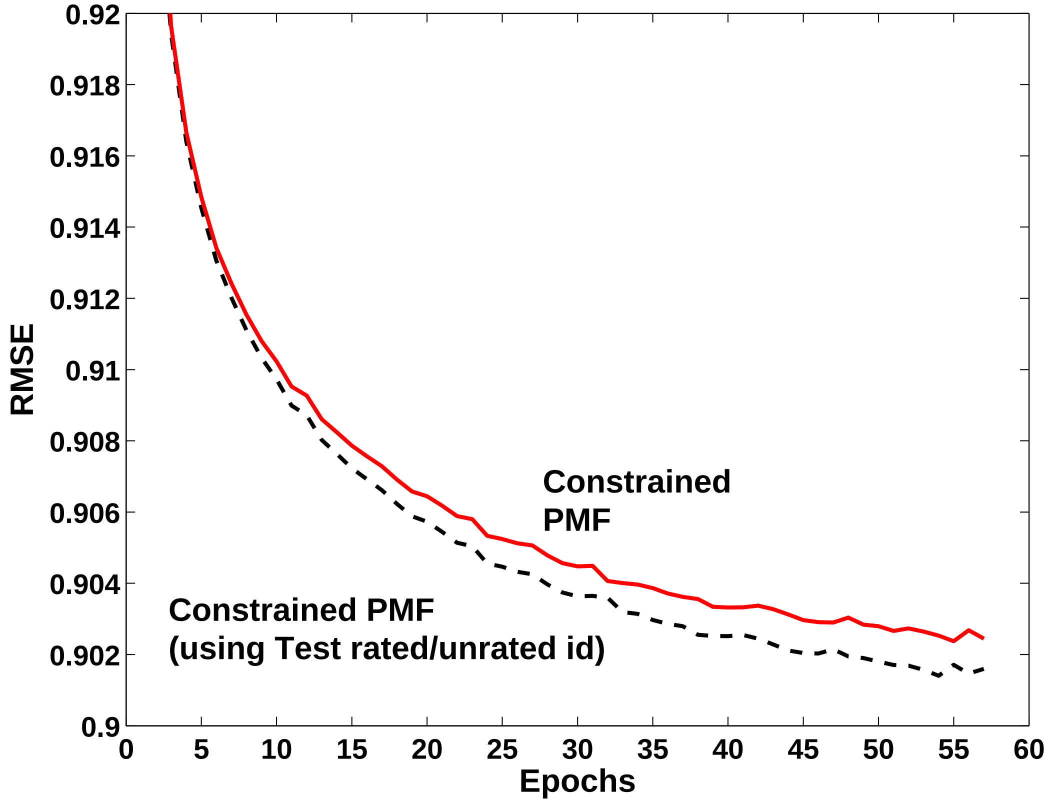




- Left Panel: Performance of constrained PMF, PMF and movie average algorithm that always predicts the average rating of each movie.

- Right panel: Distribution of the number of ratings per user in the training dataset.

Experimental Results



- Performance of constrained PMF that uses an additional rated/unrated information from the test dataset.

- Netflix tells us in advance which user/movie pairs occur in the test set.

Bayesian PMF?

- Training PMF models are trained efficiently by finding point estimates of model parameters and hyperparameters.

- Can we take a fully Bayesian approach by place proper priors over the hyperparameters and resorting to MCMC methods?

- With 100 million ratings, 0.5 million users, and 18 thousand movies?

- Initially this seemed infeasible to us due to the great computational cost of handing a dataset of this size.

Bayesian PMF!

- Bayesian PMF implemented using MCMC can be surprisingly efficient.
- Going fully Bayesian improves performance by nearly 1.5% compared to just doing MAP.

Variations on PMF

- Many variations on PMF are possible:
  - Non-negative matrix factorization.
  - Training methods: stochastic, minibatch, alternating least squares, variational Bayes, particle filtering.
  - Etc.

THE END