ISYE-7406 DATA MINING & STATISTICAL LEARNING

PROJECT

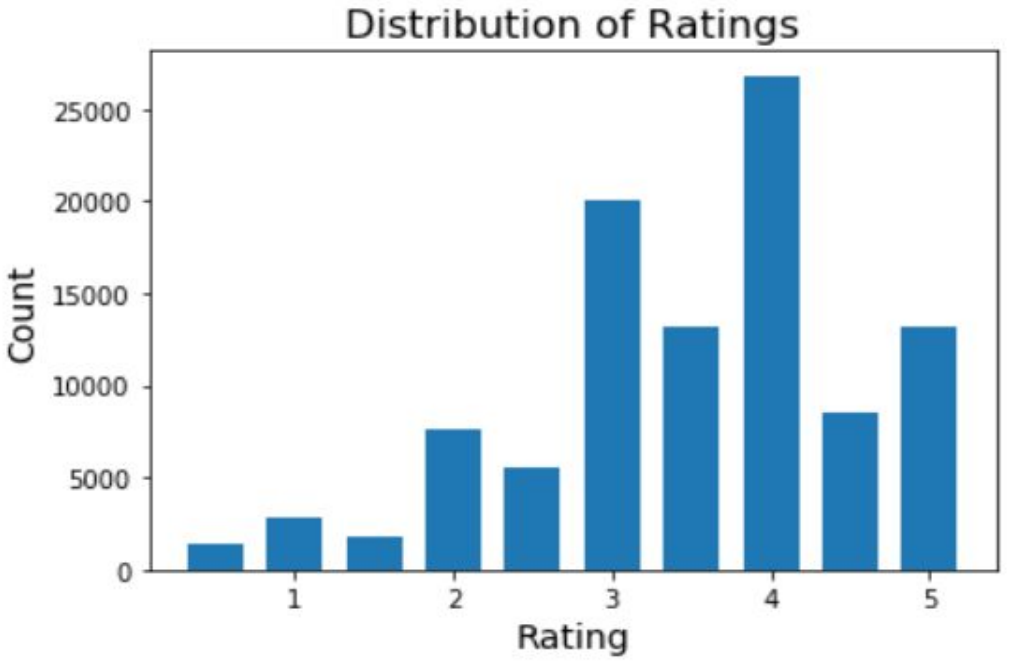
RECOMMENDER SYSTEMS – COLLABORATIVE FILTERING

VIDYUT RAO	903511348
NEVIN THOMAS	903387577
PRAVEEN MUTHUKRISHNAN	903512179

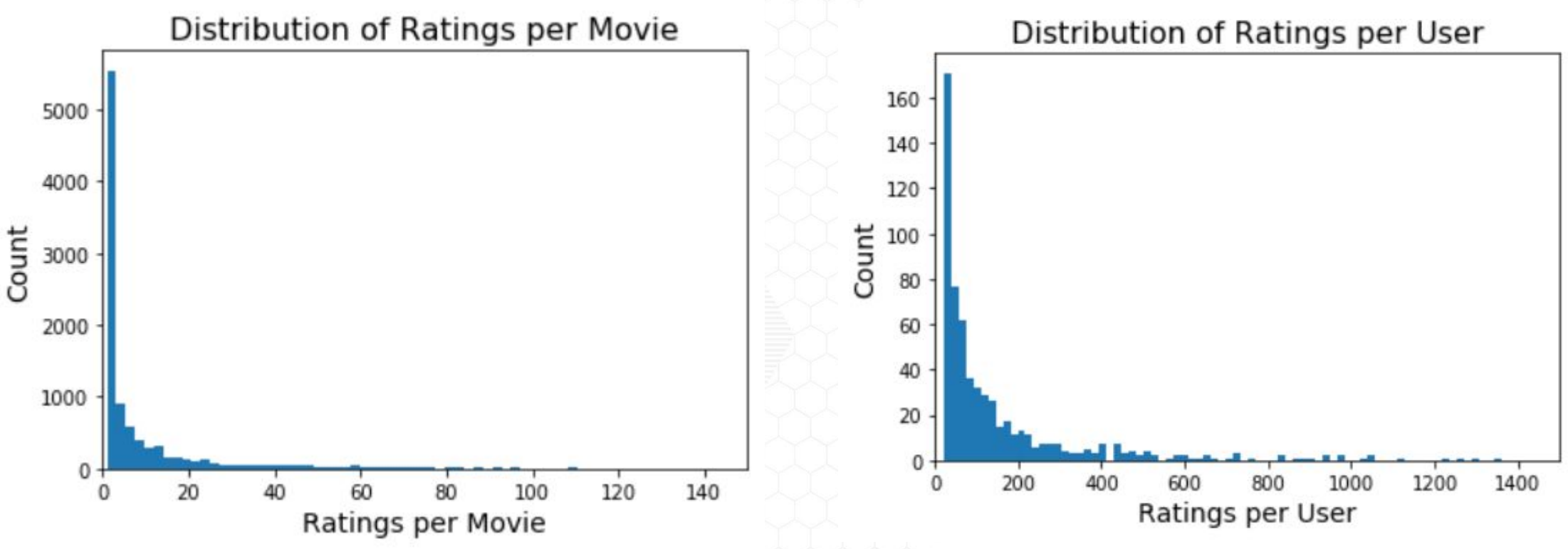
CREATING THE NEXT®

DESCRIPTION ABOUT DATA:

- This project aims to study the various collaborative filtering techniques used in recommender systems. For this purpose, the MovieLens dataset seems most appropriate
- The dataset has 100,000 ratings submitted by over 600 users, across 9000 movies. The models implemented in PyTorch, were run on our local GPUs. While the dataset isn't too large to inhibit training, it was large enough to appreciate speed ups offered GPU based computation.
- A simple pivot of the original dataset gives us the required matrix of user-item preferences
- This also gives us a large, sparse and imbalanced dataset that helps us highlight the utility of the probabilistic and bayesian treatment of matrix factorization and the other models discussed.



EXPLORATORY DATA ANALYSIS:



For smooth train-test splits, we filter out sparse movies and users before moving on to model building. The minimum number of ratings per user and movie was set to be 10.

COSINE USER-USER SIMILARITY MODEL (Baseline Model)

- It is used to measure how similar two items are irrespective of their size.
- By measuring the cosine of an angle between two vectors projected in multi-dimensional space, it can identify the similarity between users as the similarity between their rating vectors
- Since there are empty values in the matrix, we impute it by filling the mean of each user into empty values
- Ratings of similar users are then weighted with similarity score and the mean is computed
- Can be adapted to find similar items by calculating item-item similarity score

Results

- We implemented the model using the `cosine_similarity` function from the `sklearn` library
- Test RMSE obtained: **1.1105**
- We are going to treat this as the baseline accuracy throughout the study

WEIGHTED MEAN MODEL

- In the mean rating model, movies are ranked based on the mean ratings.
- Irrespective of the user information, the recommendation will be similar for all users
- This approach is ***biased*** and favours movies with fewer ratings, as movies with large number of ratings seems to be less extreme in its mean ratings
- By giving weighted score to movies, it overcomes the biased nature of mean rating model
- With weighted mean model, many good ratings outweigh few ratings
- Changing the regularization parameter determines how much weight is put on movies with many ratings
- The RMSE might decrease compared to mean rating model and so there is a tradeoff between interpretability and predictive power

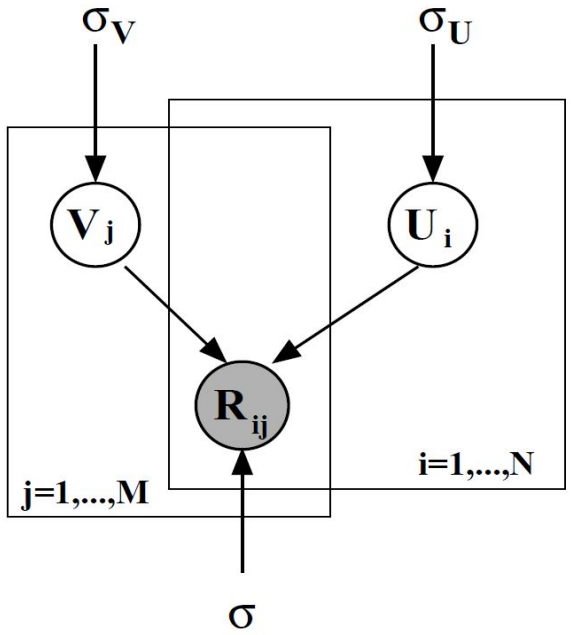
Results

- Test RMSE obtained: **0.9346**
- 15.8% improvement from the baseline model

PROBABILITY MATRIX FACTORIZATION

- PMF maps both users and movies to a joint latent factor space of dimensionality *D*, such that user-item interactions are modelled as inner products in that space
- Accordingly, for ***N*** users and ***M*** movies, the ***N * M*** preference matrix ***R*** is given by the product of ***N*D*** user coefficient matrix U^T and a ***D * M*** movie factor matrix ***V***
- Since we measure performance by RMSE, we adopt a probabilistic linear model with Gaussian observation noise
- Maximizing log-posterior over movie and user features with hyperparameters is equivalent to minimizing the sum-of-squared errors with quadratic regularization terms:

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



where,

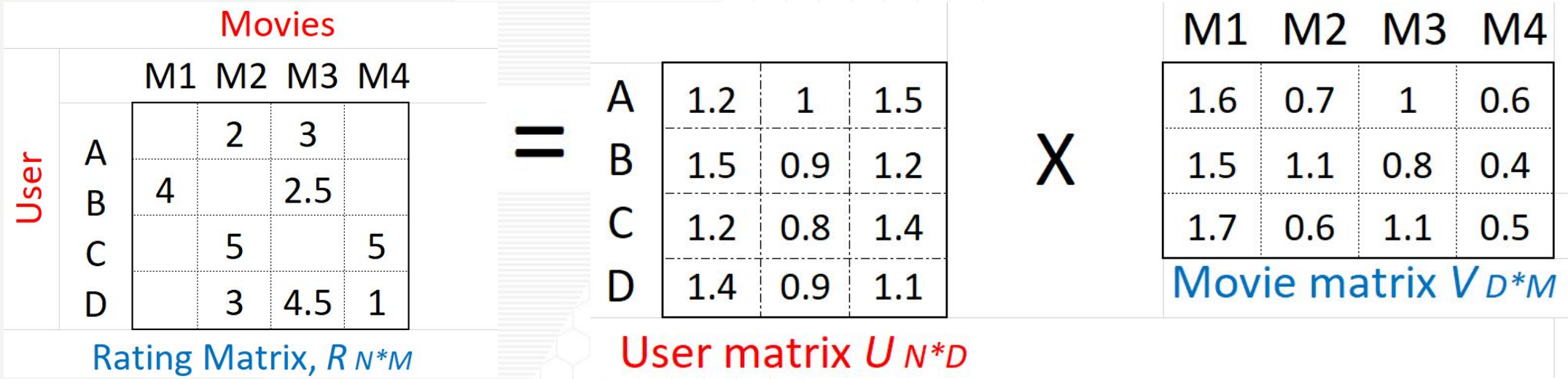
$\lambda_U = \sigma^2/\sigma_U^2$ and $\lambda_V = \sigma^2/\sigma_V^2$

$\| . \|_{Fro}$ denotes the Frobenius norm

Source : Mnih, Andriy, and Russ R. Salakhutdinov. "Probabilistic matrix factorization."

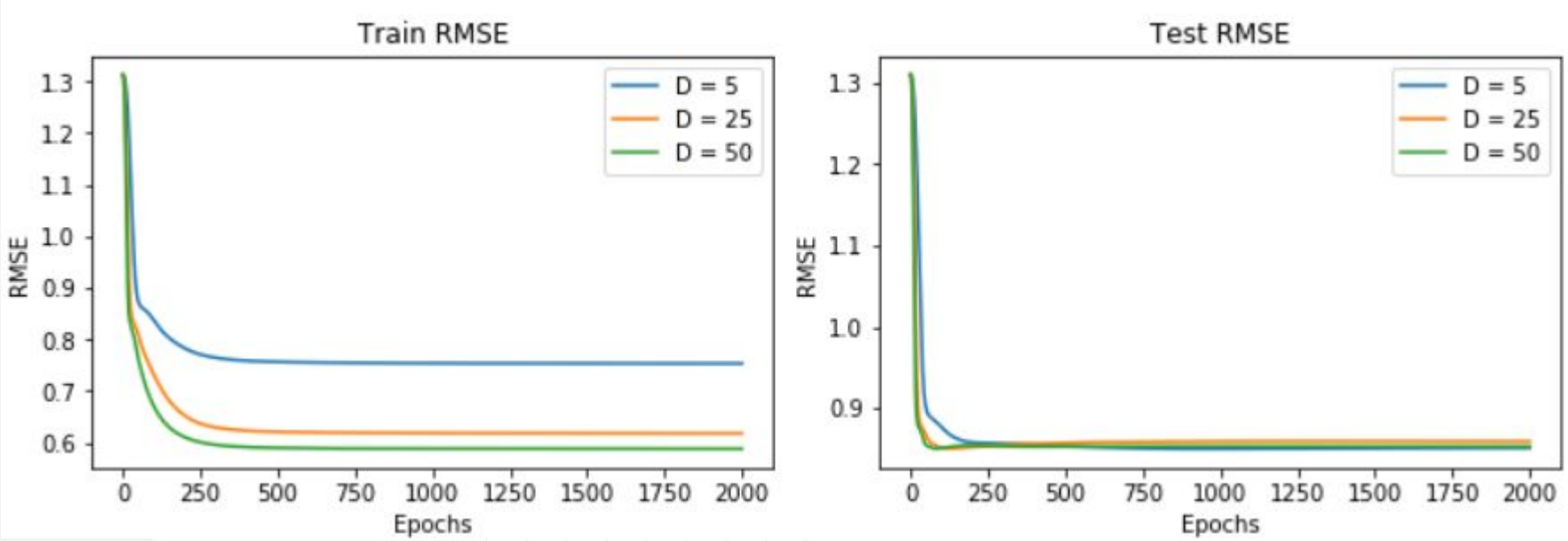
IMPLEMENTATION OF PMF

- We start with the ratings matrix R of movielens data, with users as rows and movies as columns. Then, we remove sparse users and sparse movies
- We factor R into two matrices U and V representing latent feature vectors for users and movies
- This enables us to compare users with each other and movies with each other, and predict ratings of movies users have not seen
- D represents the number of latent feature vectors which are derived from normal distribution. In the example below, we use 3 latent feature vectors



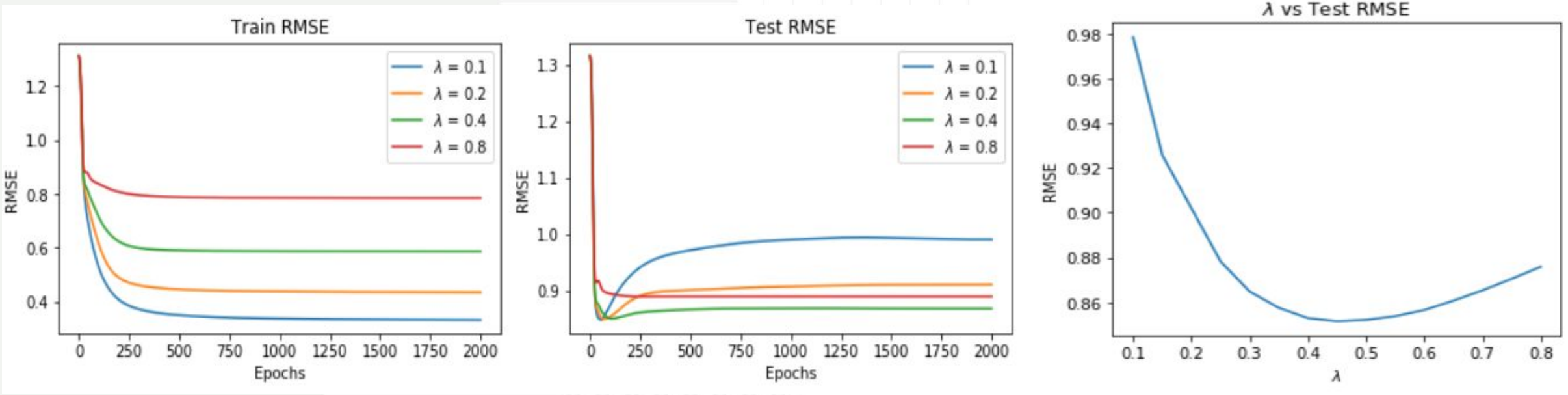
PMF RESULTS: LATENT FEATURES

- We evaluate the performance of the PMF model trained in PyTorch for different number of latent feature vectors
- The number of feature vectors in the matrix factorization corresponds to the amount of information or noise retained, which directly influences the variance in the model. Increasing the number of features captures more information and hence always decreases the training RMSE
- However, the increased variance can hinder performance on the validation set and does not correspond to better RMSE



PMF RESULTS : REGULARIZATION

- To curb the variance and noise included in the model, we add L2 regularization to the parameters in both the user and item feature matrices.
- We study the influence of λ on the training set performance. Higher values of λ always reduce variance and increase bias, thus we see a typical dip in RMSE on the training set.
- On the testing set, we see that larger λ values smooth out the RMSE vs epoch curve. Very large λ include too much bias and thus we see best performance around λ_u, $\lambda_v = 0.45$ (the bias-variance tradeoff)
- A final test RMSE of **0.8567** was obtained, a **22.85%** improvement over the baseline



BIASED PROBABILITY MATRIX FACTORIZATION

- PMF tries to capture the interaction between users and items that produce different rating values. However, the variations in these values is either associated with users or items called biases, independent of any interaction between them
- Rating values cannot be completely explained by interaction of the form : $\mathbf{U*V^T}$ where U is the user matrix and V is item matrix
- Instead, biased PMF tries to identify the portion of these values that individual user or movie biases can explain, subjecting to true interaction portion of the data to factor modelling
- As such, the observed rating is divided into three portions: item bias, user bias and user-movie interaction
- The system learns by minimizing the squared error function:

$$\min_{U,V,b^u,b^v} \sum_{i=1}^{N} \sum_{j=1}^{M} I_{ij}(R_{ij} - b_i^u - b_j^v - U_i^T V_j)^2 + \lambda_U(||U_i||^2 + b_i^{u2}) + \lambda_V(||V_j||^2 + b_j^{v2})$$

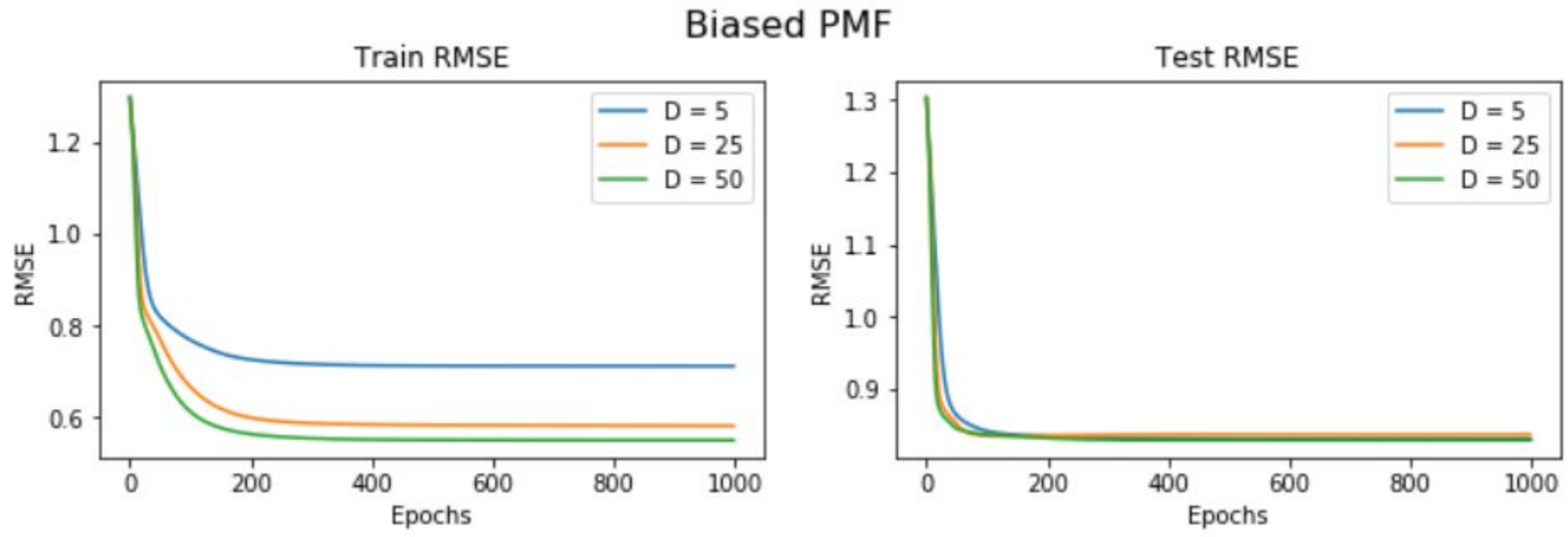
where,

b^u indicate user bias

b^v indicate movie bias

BIASED PMF : RESULTS

- Similar to the normal PMF case, we observe a general decrease in the training and test RMSE with an increase in the number of latent factors (D).
- However, we cannot generalize this trend because choosing a large D means increased variance in the model and might not result in improved test RMSE.
- Adding the bias terms significantly improved the RMSEs from the normal PMF case.

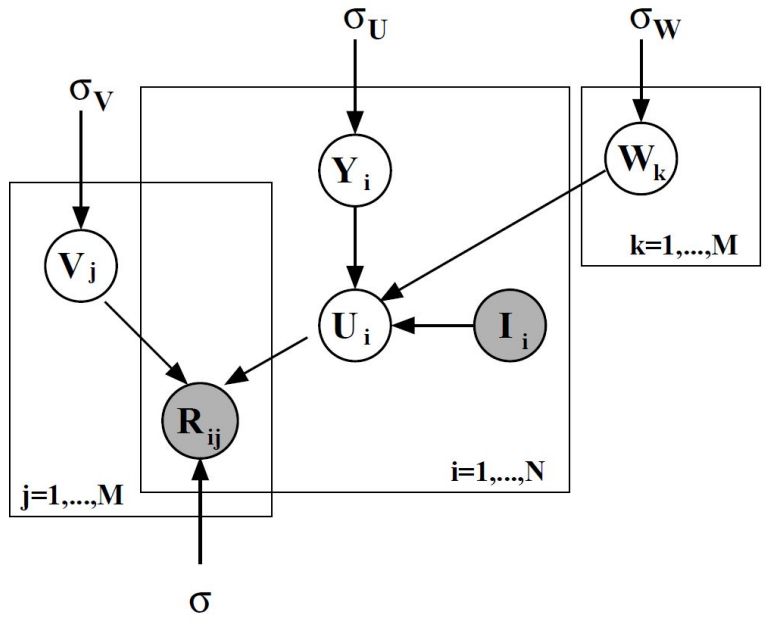


CONSTRAINED PROBABILITY MATRIX FACTORIZATION

- For the PMF model, the latent vector for an infrequent user '*i*' will be close to the average in the trained model. The predicted ratings for that user will be close to the global average ratings.
- Constrained PMF proposes a better way: find other users who watched the same items as user '*i*' and constrain the latent vectors of the similar users to be similar
- It constrains user matrix **U** with latent similarity constraint matrix W as follows:

$$U_{i,:} = Y_{i,:} + \frac{\sum_{k=1}^{M} I_{i,k} W_{k,:}}{\sum_{k=1}^{M} I_{i,k}}$$

  - **Y** captures the user offset
  - k-th row of W captures the effect that movie 'k' has on the user feature vector



- As with the PMF model, maximizing log posterior is equivalent to minimizing sum-of-squared error with quadratic regularization terms:

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij} - g([Y_i + \frac{\sum_{k=1}^{M} I_{ik} W_k}{\sum_{k=1}^{M} I_{ik}}]^T V_j))^2 + \frac{\lambda_Y}{2}\sum_{i=1}^{N}||Y_i||_{Fro}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M}||V_j||_{Fro}^2 + \frac{\lambda_W}{2}\sum_{k=1}^{M}||W_k||_{Fro}^2$$

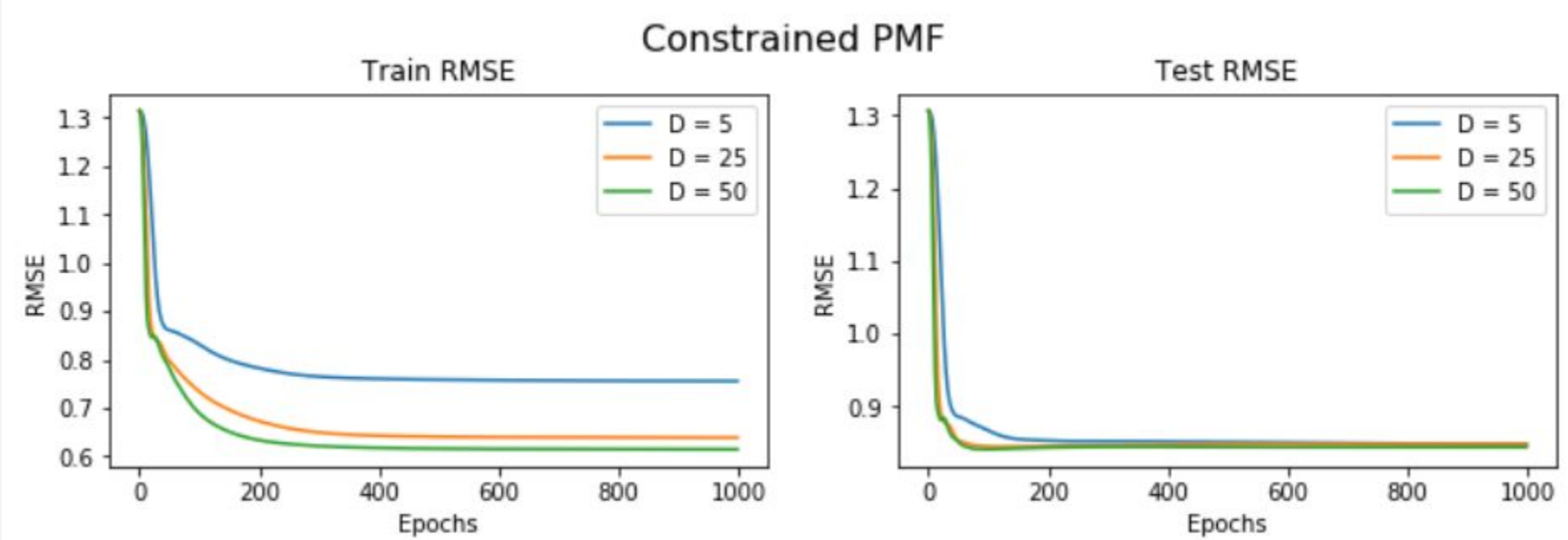
where,

$\lambda_U = \sigma^2/\sigma_U^2$, $\lambda_V = \sigma^2/\sigma_V^2$ and $\lambda_W = \sigma^2/\sigma_W^2$

Source : Mnih, Andriy, and Russ R. Salakhutdinov. "Probabilistic matrix factorization."

CONSTRAINED PMF : RESULTS

- We observe a general improvement in the test and train RMSEs as we increase the number of latent vectors involved in estimating the rating
- While the constrained version outperformed the normal PMF, it could not beat the performance of the biased PMF version



BAYESIAN PROBABILITY MATRIX FACTORIZATION

- PMF through collaborative filtering is prone to overfitting if regularization parameters are not tuned properly.
- In the BPMF model, the prior distributions of the user and movie feature matrices are assumed to be Gaussian.

$$p(U|\mu_U, \Lambda_U) = \prod_{i=1}^{N} \mathcal{N}(U_i|\mu_U, \Lambda_U^{-1}),$$

$$p(V|\mu_V, \Lambda_V) = \prod_{i=1}^{M} \mathcal{N}(V_i|\mu_V, \Lambda_V^{-1}).$$

- We then place Gaussian-Wishart priors to the user and movie hyperparameters $\theta_U = \{\mu_U, \Lambda_U\}$ and $\theta_V = \{\mu_V, \Lambda_V\}$

$$p(\Theta_U|\Theta_0) = p(\mu_U|\Lambda_U)p(\Lambda_U) = \mathcal{N}(\mu_U|\mu_0, (\beta_0\Lambda_U)^{-1})\mathcal{W}(\Lambda_U|W_0, \nu_0)$$

$$p(\Theta_V|\Theta_0) = p(\mu_V|\Lambda_V)p(\Lambda_V) = \mathcal{N}(\mu_V|\mu_0, (\beta_0\Lambda_V)^{-1})\mathcal{W}(\Lambda_V|W_0, \nu_0).$$

- Here W is the Wishart Distribution with ν_0 degrees of freedom and a D * D scale matrix W_0:

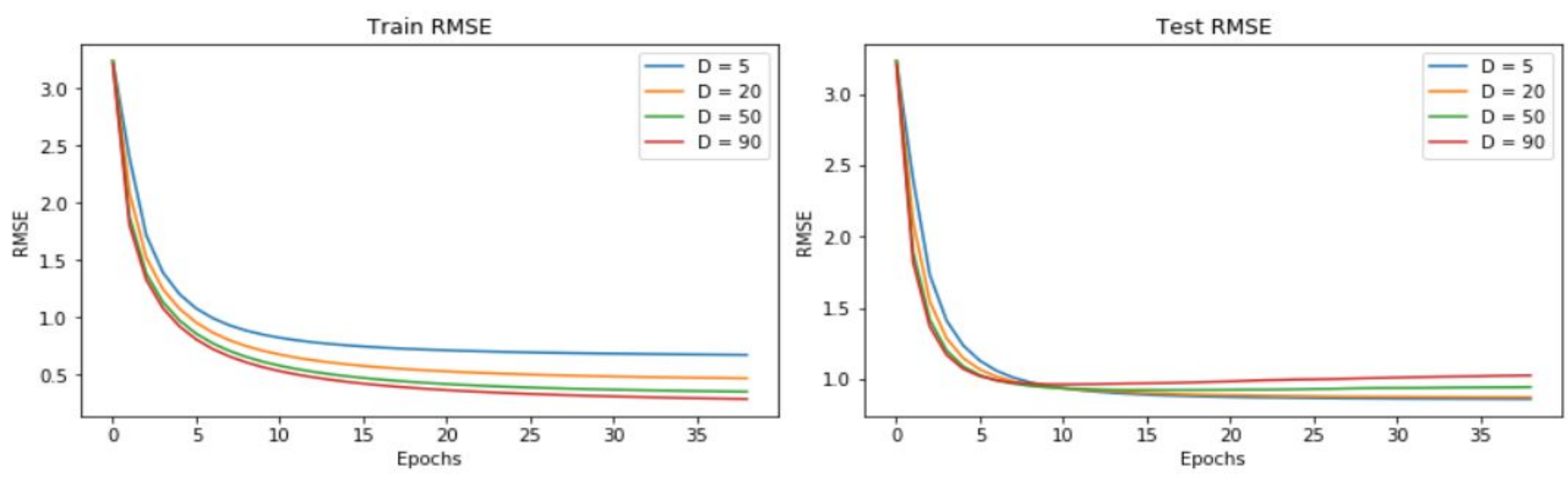
$$\mathcal{W}(\Lambda|W_0, \nu_0) = \frac{1}{C}|\Lambda|^{(\nu_0 - D - 1)/2} \exp\left(-\frac{1}{2}\mathrm{Tr}(W_0^{-1}\Lambda)\right),$$

- We then utilize the Gibbs sampling algorithm (a Markov Chain Monte Carlo algorithm) to cycle through the latent variables, sampling through each one conditional on the current value of all other variables.
- With each MCMC step, we improve on our estimate for the U and V matrices.

Source : Salakhutdinov, Ruslan, and Andriy Mnih. "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo."

BAYESIAN PMF : RESULTS

- We run Bayesian PMF for different values of D, each for 40 MCMC steps.
- As seen with previous models, larger values of D lead to more variance as can be seen in the boosted performance on the training set, but a corresponding loss in performance on the test set.
- The best results were obtained at D = 10, with test RMSE of **0.833**, a **24.99%** improvement over the baseline.



DEEP PROBABILITY MATRIX FACTORIZATION

- Exploits the predictive power of deep neural networks to give accurate rating predictions
- Instead of using a fixed dot-product as recommendation we utilized dense layers so the network can find better combinations
- Predictions are made as shown below by minimizing the following cost function

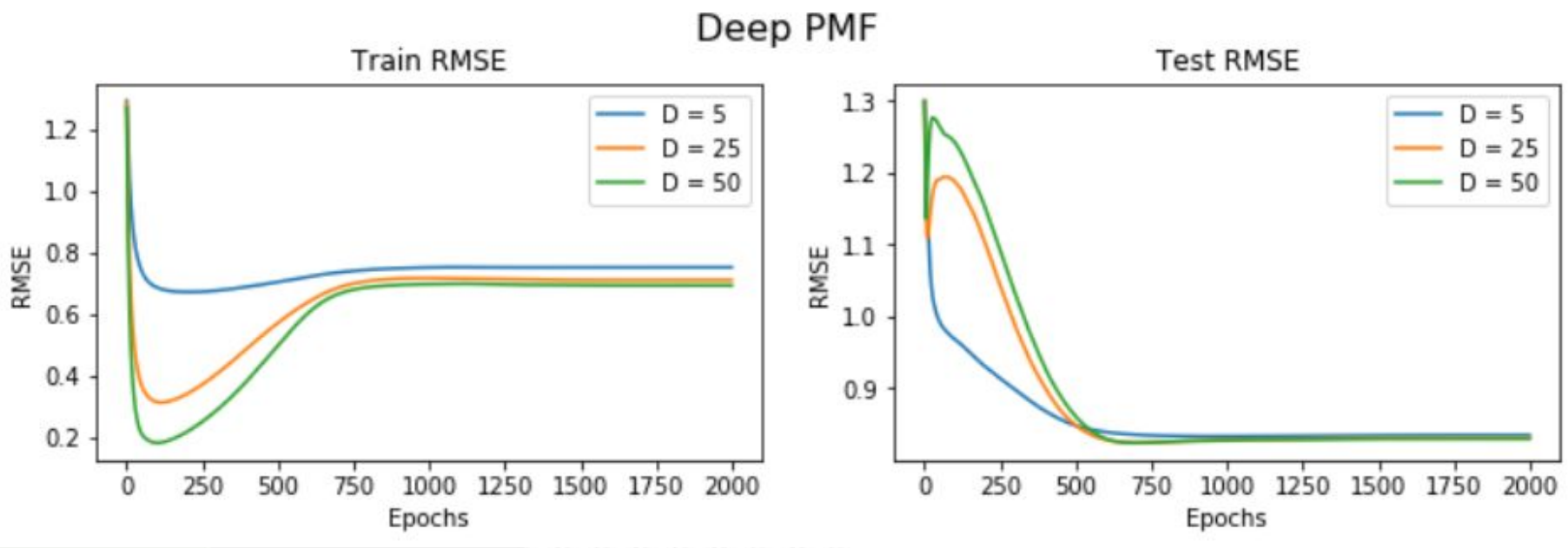
$$\hat{R} = \sigma(ReLU(\mathbf{U}^T\mathbf{V})\mathbf{L})$$

$$\min_{\mathbf{U},\mathbf{V},\mathbf{L}} ||\mathbf{I}_{nz} * (\mathbf{R}^{'} - \hat{\mathbf{R}}_s)||^2 + \lambda_u||\mathbf{U}||^2 + \lambda_v||\mathbf{V}||^2 + \lambda_l||\mathbf{L}||^2$$

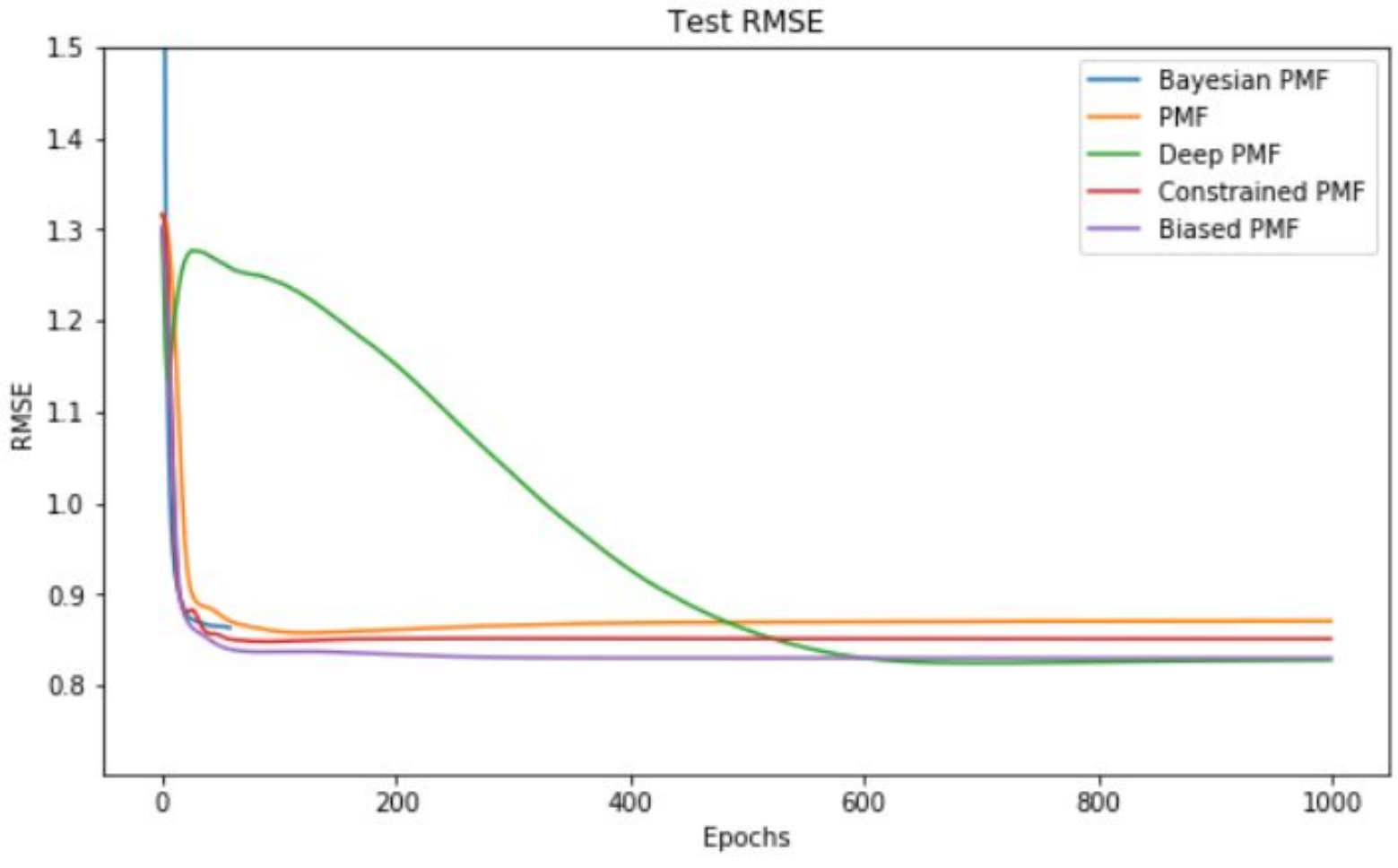
- Where σ is the sigmoid activation function, $\hat{\mathbf{R}}_s$ is the re-scaled predicted ratings matrix, $\mathbf{I}_{nz}$ is the indicator matrix with entry 1 at i^{th} row and j^{th} column if user i has rated movie j and 0 otherwise
- **L** is the (M*M) matrix which we use as the dense layer

DEEP PMF : RESULTS

- Deep PMF gave us the best RMSE on the test set harnessing the power of deep layers
- Compared to other methodologies, deep PMF took considerably more amount of time as well as number of epochs for convergence because of the large number of parameters involved.



SUMMARY


Test RMSE
RMSE
Epochs
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0
200
400
600
800
1000
Bayesian PMF
PMF
Deep PMF
Constrained PMF
Biased PMF

SUMMARY

Method	Parameters	RMSE	Performance over Baseline (%)
Weighted Mean	NA	0.9346	15.84
PMF	D=25, λ_u= 0.45, λ_v= 0.45	0.8567	22.85
Biased PMF	D=50, λ_u= 0.42, λ_v= 0.42	0.8305	25.21
Constrained PMF	D=50, λ_y=0.5, λ_w=0.5, λ_v=0.5	0.8510	23.37
Deep PMF	D=50, λ_u=0.3, λ_v=0.3, λ_l=40	0.8297	25.29
Bayesian PMF	D = 10, α = 0.9	0.833	24.99

We could conduct an extensive cross validation only for the normal PMF because of the limited computational power at our disposal. Therefore, there is room for improving the performance of the advanced models further.

CONCLUSION

- Probabilistic Matrix Factorization exceeds by far the performance of simple methodologies like cosine user-user similarity and weighted means for rating predictions.
- Among the different versions of PMF that we implemented, Deep PMF exhibited the best performance in terms of test RMSE
- Biased PMF performed almost as well as deep PMF in spite of its simplicity, pointing towards the importance of accounting for user and movie specific biases in modelling the ratings.
- The number of latents vectors should be chosen such that it can hold sufficient information about the users and movies while ensuring that the model variance is not too high.
- We observed that the values of regularization parameters have a considerable impact on the test RMSEs and therefore tuning them is important.
- Choosing the right hyperparameters (latent factors, regularization factors etc.) using cross validation might not be computationally feasible for large datasets, i.e., there is a scaling issue.
- Without the need for manually tuning the hyperparameters, the Bayesian treatment of PMF offers a scalable alternative with greater performance.

REFERENCES

[1] Andriy Mnih and Russ R Salakhutdinov. Probabilistic matrix factorization. In Advances in neural information processing systems, pages 1257–1264, 2008.

[2] Tinghui Zhou, Hanhuai Shan, Arindam Banerjee, and Guillermo Sapiro. Kernelized probabilistic matrix factorization:Exploiting graphs and side information. In Proceedings of the 2012 SIAM international Conference on Data mining,pages 403–414. SIAM, 2012.

[3] Lu Liu. Probabilistic matrix factorization for music recommendation.

[4] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. Computer, 42(8):30–37, 2009.

[5] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In Proceedings of the 25th international conference on Machine learning, pages 880–887, 2008.

[6] C'ecile Log´e and Alexander Yoffe. Building the optimal book recommender and measuring the role of book covers in predicting user ratings.