NATIONAL UNIVERSITY OF SINGAPORE

Faculty of Science

AY 2021/2022, SEMESTER 2

DSA4212: Optimisation for Large-Scale Data-Driven Inference

Assignment 2

Done by: Group 12

Chua Xin Xuan (A0205767X)

Joey Tan Xin Yi (A0206334N)

Quek Su Ning (A0205557A)

Tan Jie Yi (A0206383H)

1 Project Description

The project we chose to do is Collaborative Filtering. We aim to explore matrix factorization methods for collaborative filtering and compare it with other methodologies. In particular, we explored different optimization techniques for matrix factorisation such as the naive gradient descent, stochastic gradient descent, alternated minimization and L-BFGS. Also, we attempted other methodologies like memory based collaborative filtering, non-negative matrix factorisation and probabilistic matrix factorisation.

2 Exploratory Data Analysis

The ratings data contains 1,000,209 ratings of movies ranging from 1 to 5 by users. The UserIDs and MovieIDs range from 1 to 6040 and 1 to 3952 respectively. However, only 3706 out of the 3952 movies are rated.

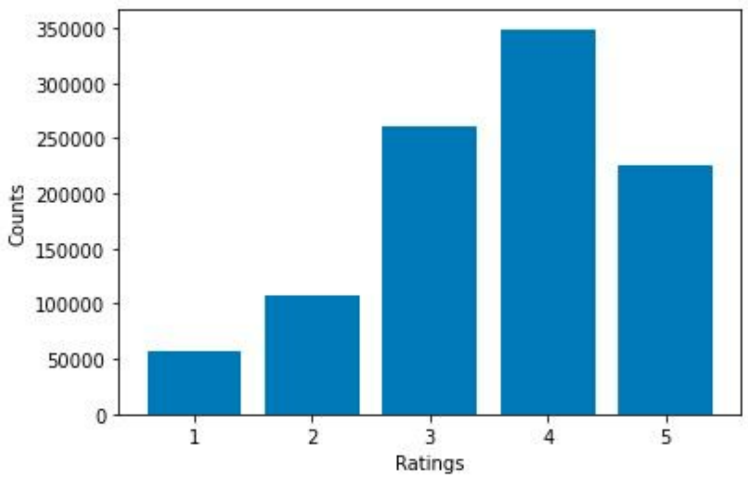


Figure 1: Bar plot of distribution of ratings

From *Figure 1,* we observe that most of the movies are given a rating of 3 and 4.

3 Models

Before the implementation of any model, 10% of the data was randomly sampled and set aside for testing. Hence, only 90% of the data was used to train and validate the models. The training dataset consists of 900188 ratings of movies while the test dataset consists of 100021 ratings of movies.

3.1 Naive Methods

We built 3 naive models as our baseline model. The first naive model predicts all ratings as the mean of all ratings in the training set. The second naive model predicts rating as the mean of the user's ratings in the training set. Lastly, the third model predicts ratings as the mean of the movie's ratings in the training set. The results obtained are shown in *Table 1* below:

	Mean Rating	**Mean User rating**	**Mean Movie Rating**
RMSE	1.12	1.03	1.20

Table 1: Table of Root Mean Squared Error (RMSE) of naive models

3.2 Matrix Factorization

To get the most accurate predictions of the actual ratings, we aim to minimize the difference between our predicted ratings and the actual ratings. However, the rating matrix is large and sparse. To reduce the complexity of the problem, we do low-rank approximation to simplify the problem into a users matrix, U and a movies matrix, V.

The ratings dataframe was firstly converted to a 6040 by 3952 matrix. We defined U as a 6040 by K matrix, and V as a K by 3952 matrix, where K is the number of latent features which will be decided via cross validation. The prediction of a movie rating will be obtained by $\hat{R}_{U,V} = \langle U, V \rangle$

We will be using different optimization techniques to minimize the loss function which is given by $L(U,V) = \frac{1}{|J|} \sum_{(a,b)\in J} \left(\langle U_a, V_b \rangle - R_{a,b}\right)^2$

For cross-validation by tuning the hyperparameters, the training data was further split into 80% for training, and 20% for validation. The training dataset consists of 720150 ratings of movies while the validation dataset consists of 180038 ratings of movies.

3.2.1 Naive Gradient Descent

We attempt the Naive Gradient Descent as a baseline model for our gradient descent methods. For each iteration, we performed gradient updates on U and V such that:

$$\begin{cases} [U]_{n+1} = [U]_n - \eta \nabla L_U(U,V) \\ [V]_{n+1} = [V]_n - \eta \nabla L_V(U,V) \end{cases}$$

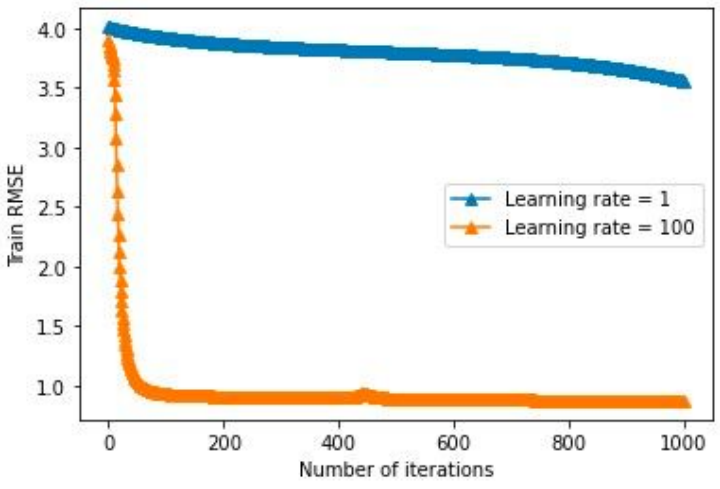


Figure 2: Plot of train loss of Naive Gradient Descent for different learning rates

From *Figure 2*, we observe that naive gradient descent methods require a high learning rate. Using a standard learning rate of 1, the loss function will need significantly more iterations to converge.

Hence, we performed cross-validation on a range of number of latent features K from 1 to 50, with a learning rate of 100 and 1000 iterations.

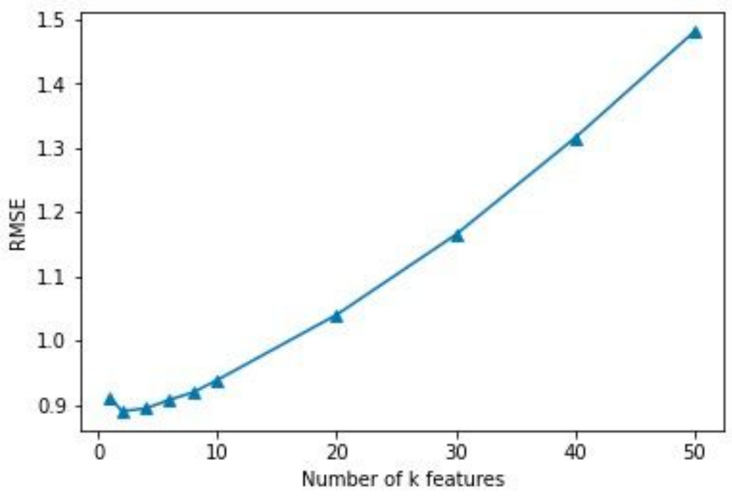


Figure 3: Cross-validation for Naive Gradient Descent

From *Figure 3*, the lowest validation RMSE of 0.890 was obtained when the number of latent features K is 2.

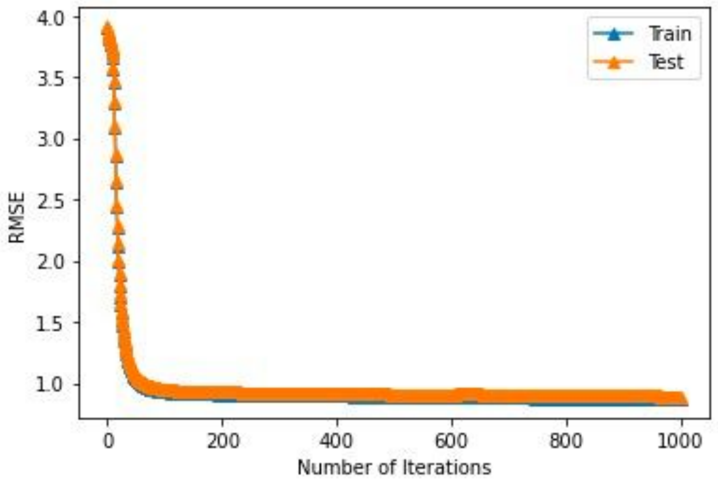


Figure 4: Plot of train & test loss of Naive Gradient Descent when K = 2

From our above results, training the naive gradient descent model on the entire training data yields a test RMSE of 0.891.

3.2.2 Stochastic Gradient Descent

We explored Stochastic Gradient Descent methods to compare if it does better than the naive method. Furthermore, given the large matrix size, Stochastic Gradient Descent would allow a faster convergence compared to Naive Gradient Descent where all training samples are used at each iteration. For each minibatch of each epoch, we performed the following gradient updates:

$$\begin{cases} [U]_{n+1} = [U]_n - \eta \widehat{\nabla L}_U(U, V) \\ [V]_{n+1} \ = [V]_n - \eta \widehat{\nabla L}_V(U, V) \end{cases}$$

We also explored Stochastic Gradient Descent with momentum which will help to accelerate the gradient descent, resulting in a faster convergence. The following gradient updates are performed:

$$\begin{cases} [U]_{n+1} = [U]_n - \eta \widehat{\nabla L}_U(U, V) + \beta * \left(U_n - U_{n-1}\right) \\ [V]_{n+1} \ = [V]_n - \eta \widehat{\nabla L}_V(U, V) + \beta * \left(V_n - V_{n-1}\right) \end{cases}$$

Both Stochastic Gradient Descent and Stochastic Gradient Descent with Momentum were performed using the following parameters: minibatch_size = 1000, 15 epochs and learning rate 1. The β in Stochastic Gradient Descent with Momentum was defined as 0.9.

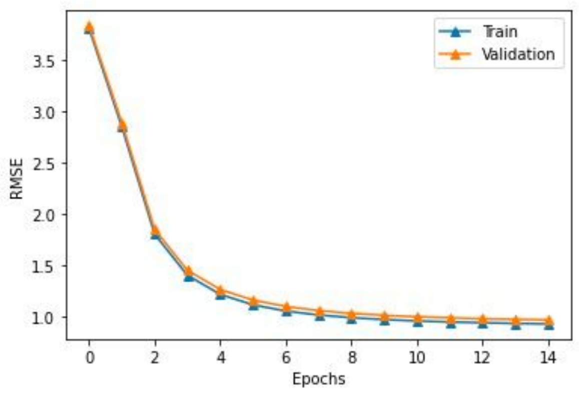


Figure 5: Plot of train & validation RMSE vs epochs for Stochastic Gradient Descent

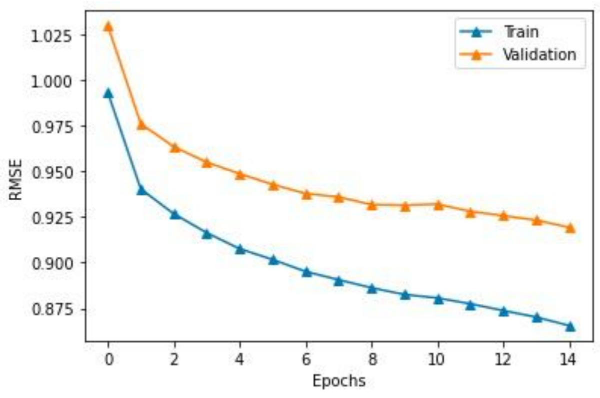


Figure 6: Plot of train & validation RMSE vs epochs for Stochastic Gradient Descent with momentum

From *Figure 5*, we observe that Stochastic Gradient Descent with Momentum was able to achieve a much lower validation RMSE of 0.919 with the same learning rate and epoch, compared to Stochastic Gradient Descent with a validation RMSE of 0.965.

Hence, we will be using Stochastic Gradient Descent with Momentum for subsequent analysis. However, from *Figure 6*, a bigger batch size and learning rate would be required for it to converge.

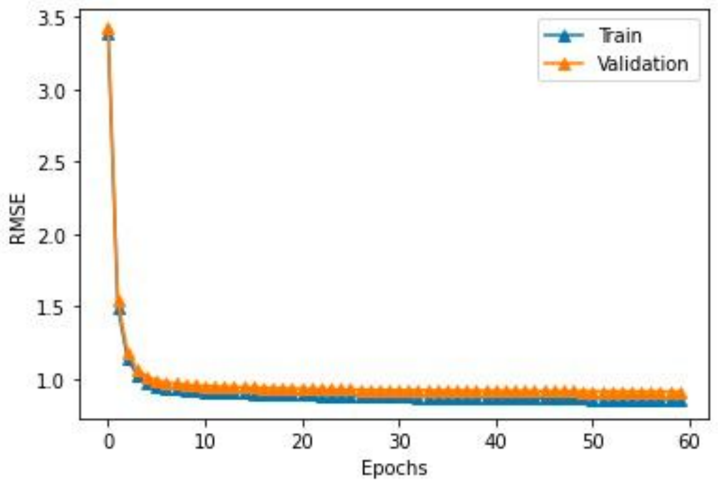


Figure 7: Plot of train & validation RMSE vs epochs for Stochastic Gradient Descent with momentum

From *Figure 7*, we note that a minibatch_size of 5000, 60 epochs, learning rate 1 and β of 0.9 was sufficient for Stochastic Gradient Descent with Momentum to converge. The validation RMSE was 0.907. We will use the same parameters for cross-validation to determine the optimal number of latent features, K.

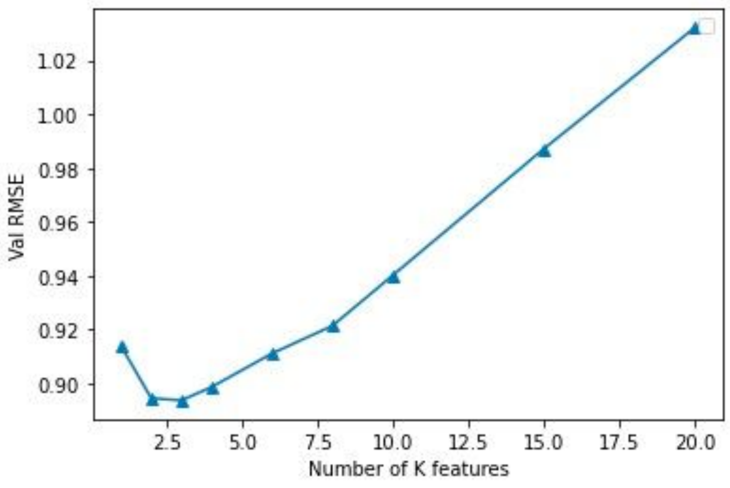


Figure 8: Cross-validation for Stochastic Gradient Descent with momentum

From our cross-validation in *Figure 8*, we achieved the lowest validation RMSE of 0.894 when the number of latent features, K, is 3.

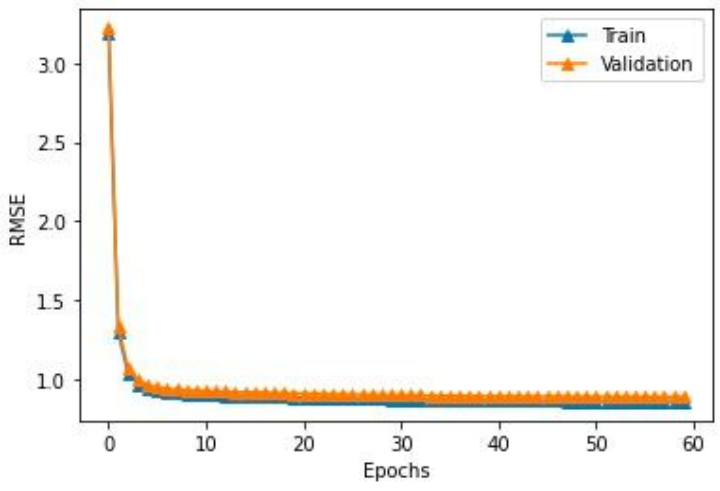


Figure 9: Plot of train & test loss of Stochastic Gradient Descent with momentum when K = 3

From our above results, training the Stochastic Gradient Descent with Momentum model on the entire training data yields a test RMSE of 0.891, which was similar to what we obtained from the Naive Gradient Descent model, but at a much smaller learning rate.

3.2.3 Alternated Minimization

We now explore the use of alternated minimization as an alternative to naive and stochastic gradient descent. Alternated minimization [2] alternates the minimization of the users and the movies matrix. Figure 10 [3] describes the steps for each iteration in alternated minimization. To summarize, for each iteration, we first fix the vectors associated with users and optimize to find the best set of vectors associated with movies. Then, we fix the vectors associated with movies and optimize to find the best set of vectors associated with users. Alternated minimization iterates the steps until it reaches a stopping criterion. For each iteration we perform the following update:

$$\begin{cases} \text{Fix } U\text{, and update } V \text{ by taking (one or more) gradient descent steps on the function } U \mapsto L(U,V); \\ \text{Fix } U\text{, and update } V \text{ by taking (one or more) gradient descent steps on the function } V \mapsto L(U,V)\,. \end{cases}$$

As mentioned previously, we set aside 20% of the training data as our validation set and use it to tune hyperparameters in alternated minimization.

Firstly, we used cross validation to determine the optimal number of gradient descent steps to optimize each matrix. A range of iterations were considered, namely 1, 5, 10, 20 and 40. 1 gradient descent iteration suggests that the movies matrix was optimized only once for each alternated minimization step and likewise for the users matrix. On the other hand, 40 gradient descent iterations suggest that the values for each matrix were updated 40 times for each alternated minimization step. Alternated minimization ran over 200 iterations using a learning rate of 20. The number of latent features, K, used was 10.

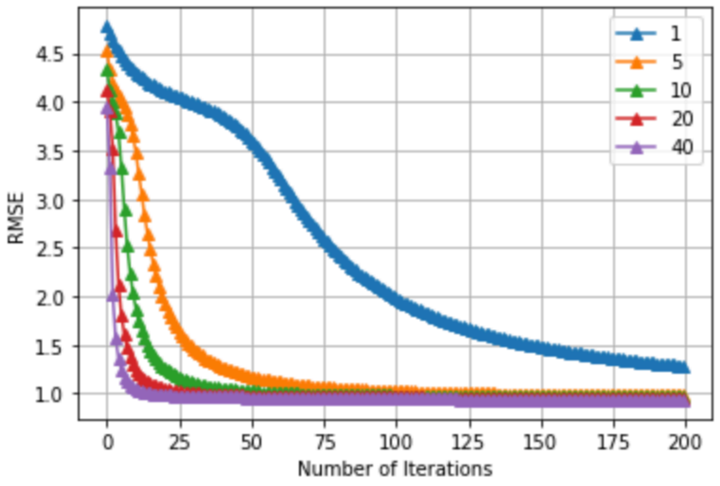


Figure 10: Validation RMSE with different number of gradient descent iterations

As observed from *Figure 10*, having more gradient descent iterations to optimize each matrix allows us to converge quickly to a low validation RMSE in fewer alternated minimization iterations. When 40 gradient descent steps were used to optimize both the users and the movies matrices, the lowest validation RMSE of 0.933 was obtained. However, the alternated minimization model also took significantly more time to train when more gradient descent iterations are used but the decrease in validation RMSE was not significant. Using 10 gradient descent steps achieved a minimum validation RMSE of 0.963 which does not differ significantly from the RMSE of 0.933 achieved using 40 gradient descent steps while taking time complexity into consideration. Furthermore, a slightly higher loss due to fewer gradient descent iterations may potentially be overcome by having more alternated minimization iterations and using a larger learning rate. Hence, for each alternated minimization iteration, we chose to use 10 gradient descent iterations to optimize the users and movies matrix for subsequent analysis.

Next, we explore the optimal number of latent features, K. Alternated minimization was run using latent features ranging from 1 to 20 over 200 iterations. For each iteration, gradient descent was performed on the users and movies matrices respectively over 10 iterations and with a learning rate of 100.

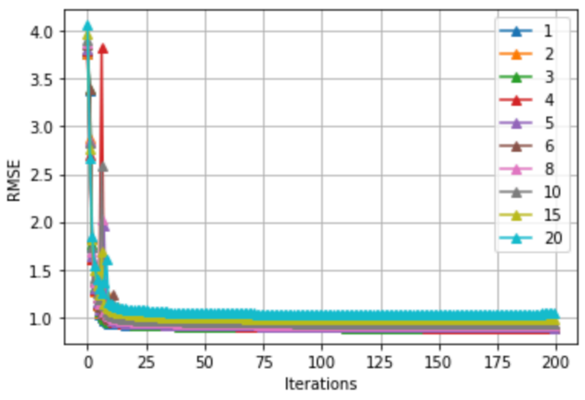


Figure 11: Validation RMSE with different number of latent features, K

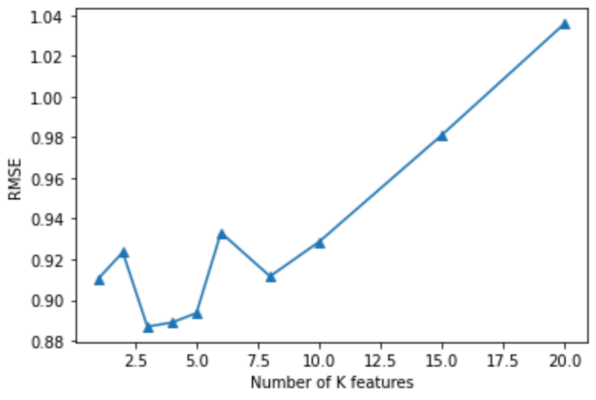


Figure 12: Minimum validation RMSE with different K

As shown in *Figures 11* and *Figure 12*, the minimum validation RMSE 0.887 was achieved when the number of latent features, K, is 3. As such, we select the number of latent features to be 3.

To find the optimal learning rate, we ran alternated minimization using different learning rates, namely 1, 10, 50, 100 and 150. For each learning rate explored, alternated minimization ran over 200 iterations with 3 latent features. For each iteration, 10 gradient descent steps were used to optimize the users and movies matrix.

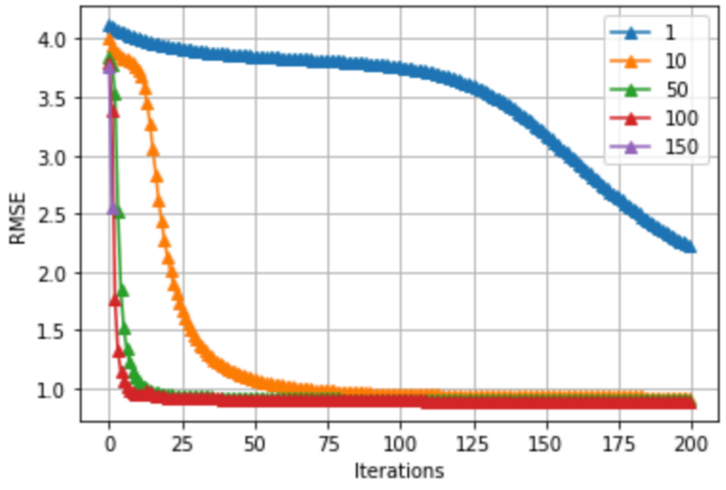


Figure 13: Validation RMSE with different learning rate

As shown in *Figure 13*, a higher learning rate generally leads to a faster convergence of the loss function. However, when the learning rate was 150, the loss function diverged by the 3rd iteration. This suggests that the learning rate was too high so we did not consider using a learning rate of 150 or higher. We decided to use a learning rate of 100 as it resulted in the lowest validation RMSE of 0.887 among the other learning rates explored.

Finally, we perform regularization on the MSE as the model has many parameters which makes it prone to overfitting. To mitigate overfitting, we included a regularization term in the loss function to be minimized. Instead of minimizing the MSE, we minimize:

$$L(U,V) = \frac{1}{|J|} \sum_{(a,b)\in J} \left(\langle U_a, V_b\rangle - R_{a,b}\right)^2 + \frac{\lambda}{|N_u|}\|U\|^2 + \frac{\lambda}{|N_v|}\|V\|^2$$

N_u and N_v refers to the number of users and movies respectively. Once again, we used the cross validation approach to choose a suitable regularization parameter, $\lambda > 0$. A range of lambda values was explored, namely 0, 0.01, 0.1, 0.5, 1 and 5. When the regularization parameter has a value of 0, it suggests that no regularization was performed as the penalty has no effect on the loss.

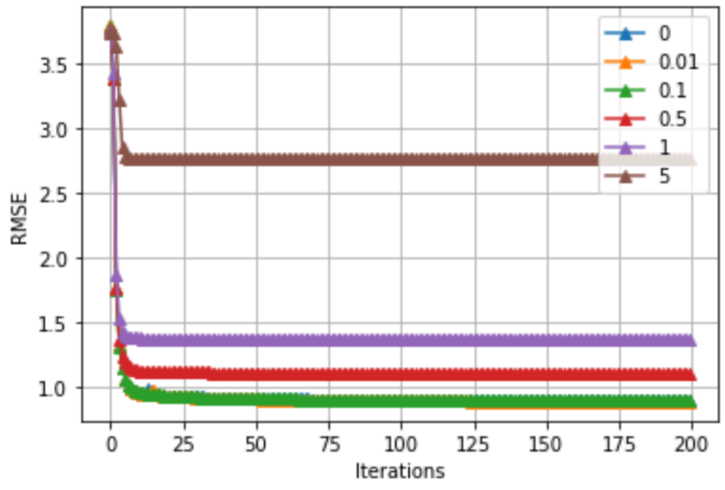
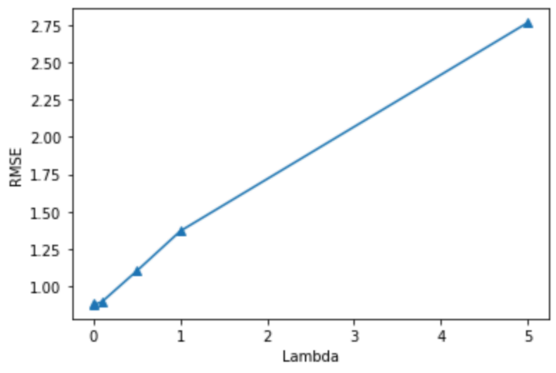


Figure 14: Validation RMSE with different regularization parameters



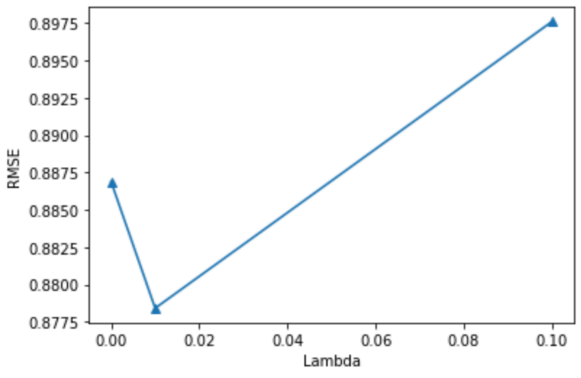


Figure 15(a): Minimum validation RMSE with different regularization parameters

Figure 15(b): Minimum validation RMSE with different regularization parameters (zoomed-in)

As shown from *Figures 14* and *Figure 15*, using a regularization parameter λ of value 0.01 resulted in the best validation RMSE value of 0.878. Hence, we chose the regularization parameter to be 0.01.

After performing cross validation to select a value for each hyperparameter, we decided to take 10 gradient descent steps to update each of the users and movies matrix while keeping the other matrix fixed. Moreover, we also selected the number of latent features to be 3, the learning rate to be 100 and the regularization parameter value to be 0.01. After finalizing the values of the hyperparameters, we trained the alternated minimization model on both our train and validation dataset over 800 iterations and then tested the model on the test set.

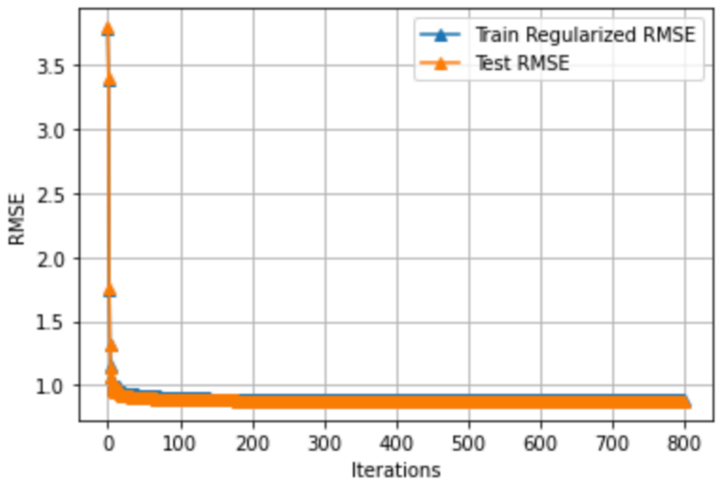


Figure 16: Loss history for final alternated minimization model

As observed in *Figure 16*, the loss function has converged. A test RMSE of 0.874 was obtained using alternated minimization.

3.2.4 L-BFGS

The computation of Hessians can be expensive in high-dimensional settings. As our project deals with high dimensions, we also explored L-BFGS, which does not compute the exact Hessians but instead, uses gradients that are computed throughout the algorithm to approximate the Hessian. L-BFGS is considered instead of BFGS as it avoids the need to store the approximations of the Hessians, making it an even faster approach.

3.2.4.1 Non-Negative Matrix Factorisation

Since ratings are positive integers, non-negative matrix factorisation is considered. Cross validation is performed to find an optimal number of features. The validation RMSE is plotted for each number of latent features as shown below in *Figure 17*:

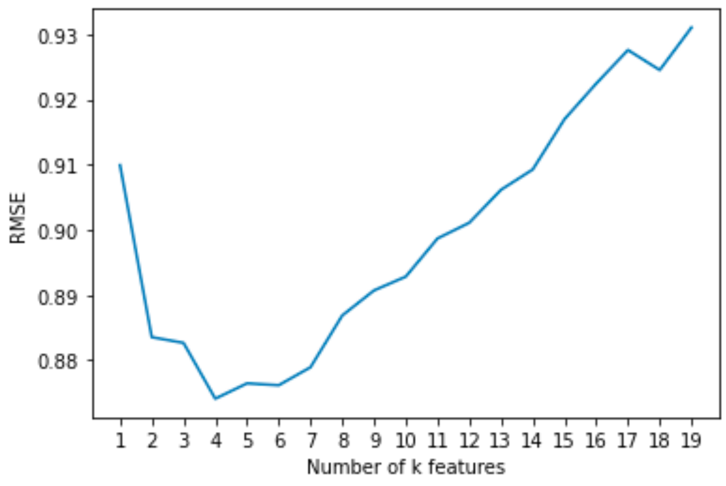


Figure 17: Validation RMSE with different number of latent features, K

From *Figure 17*, we can observe that using 4 features has achieved the lowest validation RMSE of 0.874. We set the number of latent features as 4 in our final non-negative matrix factorisation model. The plot of loss history against time history is shown below in *Figure 19*:

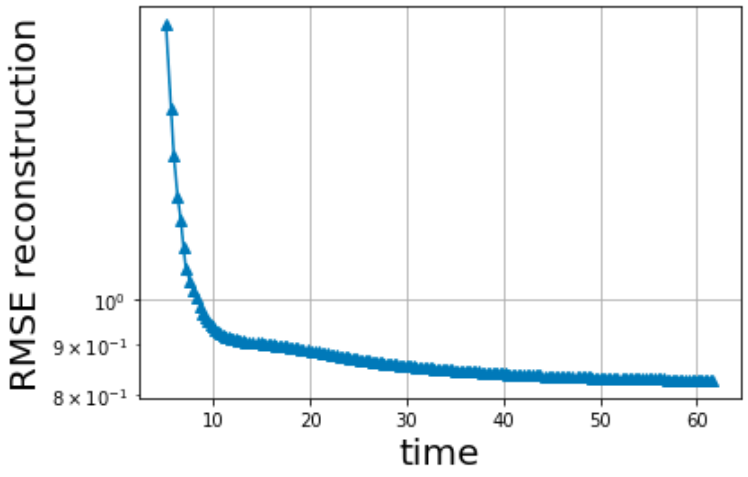


Figure 18: Loss history for final non-negative matrix factorisation model using L-BFGS

About 60 seconds was taken for the loss value to converge. A test RMSE of 0.873 was achieved by this model.

3.2.4.2 Probabilistic Matrix Factorisation

Upon research [4], many methods in collaborative filtering are unable to tackle large and sparse datasets. The Probabilistic Matrix Factorisation (PMF) model is presented as a model that could perform well in such settings.

The PMF model aims to minimize the following objective function:

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

where I_{ij} is an indicator function that indicates whether user i has rated film j, R_{ij} is the rating that user i has given film j, U_i is the vector for user i and V_j is the vector for film j. The property $||X||^2_{Fro} = X^T X$ is used in our code.

As mentioned in the paper [4], cross validation on all possible combinations of λ_u and λ_v is computationally expensive. Hence, instead of considering the various combinations of λ_u and λ_v, we referenced the various λ_u and λ_v used in the paper, and fixed both values to be the same. The values we tried are 10^{-2}, 10^{-3} and 10^{-4}.Cross validation is used to find an optimal number of features instead. Taking time complexity into account, we used L-BFGS instead of the other gradient descent methods. The plots of the validation RMSE achieved across the different number of features is shown in *Figure 19* below:

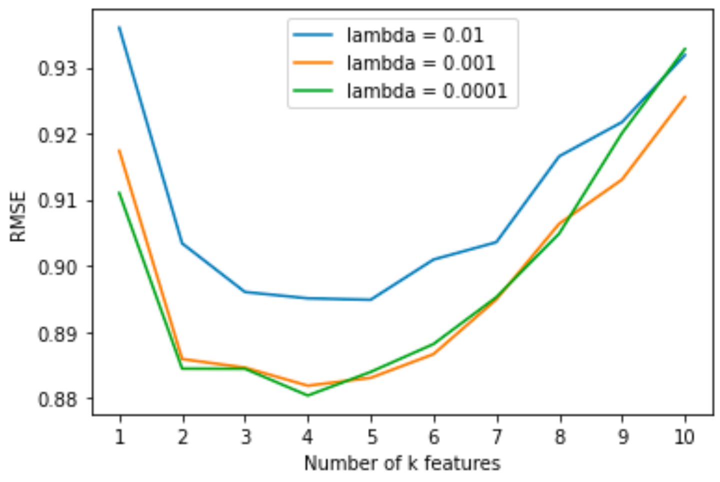


Figure 19: Validation RMSE with different number of latent features, K

Hence, in our final PMF model, we chose to set the number of latent features as 4, λ_u and λ_v to be both 10^{-4}. The plot of loss history against time history is shown below in *Figure 20*:

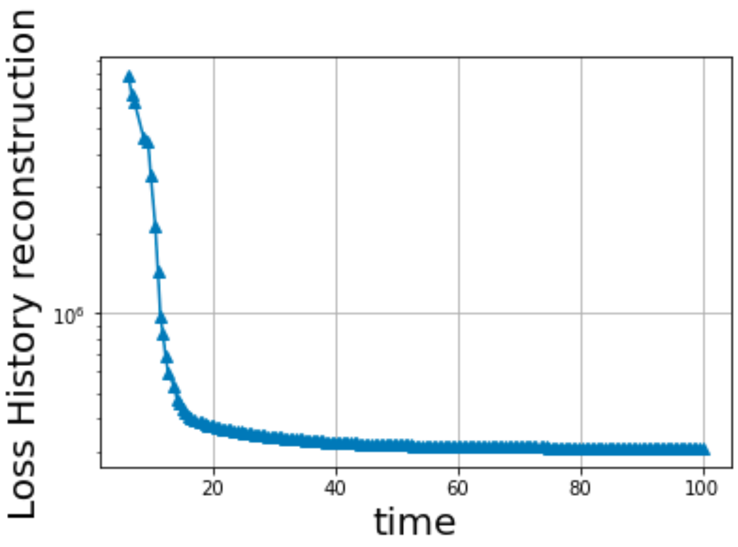


Figure 20: Loss history for final probabilistic matrix factorisation model using L-BFGS

A test RMSE of 0.877 is observed. In the paper [4], the researchers intentionally mapped their ratings and predictions using $t(x) = \frac{x-1}{K-1}$ where K is the maximum rating and $g(x) = \frac{1}{1+exp(-x)}$ respectively to a range between 0 to 1.

We plot the histogram of predictions on the test set to observe the range of predictions as shown in *Figure 21* below:

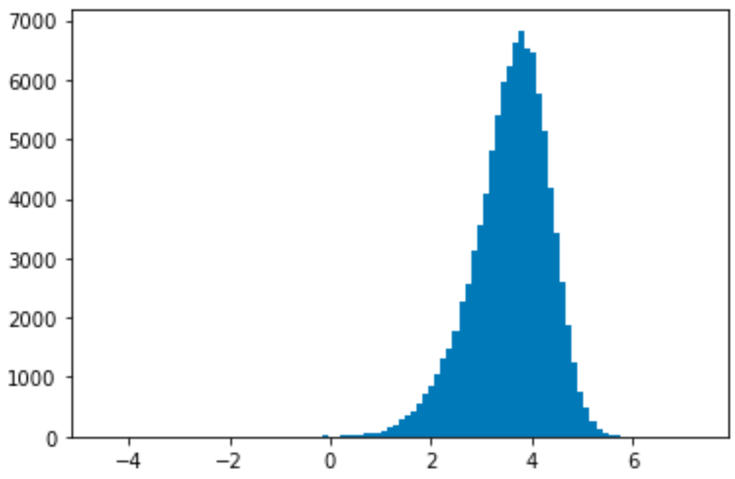


Figure 21: Distribution of predictions

While most of the range falls in the 1 to 5 range, we see some negative ratings and ratings that are slightly higher than 5. As we would like to keep our ratings between 1 to 5 for interpretation purposes, we did not use $t(x)$ and $g(x)$. Instead, we mapped values below 1 to 1 as these are values that are very low, indicating that the users are unlikely to like the film, and values above 5 to 5 for a similar reason. A new test RMSE of 0.873 is achieved.

3.3 Memory-based Approach

There are two different types of Memory-based Filtering [5], namely user based and item based. User based algorithms search for like-minded individuals and assume that similar users like similar items. On the other hand, item based algorithms search for items rated similarly by various users and assume that similar items are liked by similar users.

We explored several metrics to measure the similarity. The formula for each metric is shown below:

	User Based	**Item Based**
Cosine Distance	$s(u,v) = \sum_i \frac{r_{ui}r_{vi}}{\sqrt{\sum_i r_{ui}^2 \sum_i r_{vi}^2}}$	$s(i,j) = \sum_u \frac{r_{iu}r_{ju}}{\sqrt{\sum_u r_{iu}^2 \sum_u r_{ju}^2}}$
Euclidean distance	$s(u,v) = \sqrt{\frac{\sum_i (r_{ui} - r_{vi})^2}{\lvert I_{uv} \rvert}}$	$s(u,v) = \sqrt{\frac{\sum_i (r_{ui} - r_{vi})^2}{\lvert U_{uv} \rvert}}$
Pearson Correlation	$s(u,v) = \frac{\sum_i (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_i (r_{ui} - \bar{r}_u)^2 \sum_i (r_{vi} - \bar{r}_v)^2}}$	$s(i,j) = \frac{\sum_u (r_{ui} - \bar{r}_i)(r_{ui} - \bar{r}_j)}{\sqrt{\sum_u (r_{ui} - \bar{r}_i)^2 \sum_u (r_{ui} - \bar{r}_j)^2}}$

We predict the rating of a user for a movie by using the similarity calculated as weights.

For user based approach: $\hat{r}_{ui} = \frac{\sum_v s(u,v) r_{vi}}{\sum_v \lvert s(u,v) \rvert}$

For item based approach: $\hat{r}_{ui} = \frac{\sum_j s(i,j) r_{ju}}{\sum_j |s(i,j)|}$

	Cosine Distance	Euclidean Distance	Pearson Correlation
User Based	0.957	1.01	0.950
Item Based	0.974	1.09	**0.940**

Table 2: Test RMSE when different similarity metrics is used

As shown in Table 2, the best test RMSE obtained is 0.940 using item based approach and Pearson Correlation as the similarity matrix.

3.3.1 Mean Centering

However, one of the potential problem when using the approach above is that different users has different standards. For example, a user may rate a good movie 4 while another might rate it 5. Hence, we will deploy mean scaling algorithm which remaps a user's rating by subtracting the mean of all the rating of that particular user. By doing so, positive values represents above average ratings while negative represents below average ratings and zero will represent average rating.

The formula used to predict ratings now becomes:

For user based approach: $\hat{r}_{ui} = \bar{r}_u + \frac{\sum_v s(u,v)(r_{vi} - \bar{r}_v)}{\sum_v |s(u,v)|}$

For item based approach: $\hat{r}_{ui} = \bar{r}_i + \frac{\sum_j s(i,j)(r_{ju} - \bar{r}_j)}{\sum_j |s(i,j)|}$

	Cosine Distance	Euclidean Distance	Pearson Correlation
User Based	0.769	0.944	0.828
Item Based	**0.756**	1.09	0.799

Table 3: Test RMSE for normalized ratings when different similarity metrics is used

As expected, the test RMSE is lower if we apply mean centering to the ratings. The best test RMSE obtained is 0.756. This is achieved when the ratings are centered and cosine distance is used to measure the similarity.

4 Conclusion

To conclude, we tried a range of matrix factorization methods for collaborative filtering and other methodologies like memory based collaborative filtering. *Table 4* summarises the different methods we have explored and their performance on the test set.

Methods	Test RMSE
Naive Gradient Descent	0.891
Stochastic Gradient Descent with momentum	0.891
Alternated Minimization	0.874
Non-Negative Matrix Factorisation L-BFGS	0.873
Probabilistic Matrix Factorisation L-BFGS	0.873
Memory Based Approach	0.755

Table 4: Test RMSE achieved by the models

As shown in *Table 4*, all of the models achieved a significantly lower test RMSE as compared to the simple baseline models illustrated in *Section 3.1*.

Our best model is the memory based approach. Even though the memory based approach is a much simpler approach than matrix factorisation, it has the lowest test RMSE, which is rather unexpected. Other than achieving the lowest RMSE, the memory based approach also takes a substantially shorter time to compute, giving it an edge over matrix factorization.

5 Future Work

Thus far, our matrix factorization methods randomly initialized the users matrix, U and the movies matrix, V. Thereafter, we performed cross-validation to find the optimal number of K latent features for prediction. However, our methodologies have yet to make use of the movie's genre and user's demographics such as age, occupation, gender etc. to integrate into our models. Including this additional information may improve the prediction accuracy of our matrix factorization models as they are likely to provide some important features for the models.

Upon further research into this topic, we read the Hybrid Recommenders report [6] which details how the use of metadata awareness was incorporated into latent factor models and its effectiveness. The author compared various models in the literature and showed that incorporating metadata led to an improvement in prediction accuracy. Hence, we believe that this would be an interesting and effective improvement to our models for us to explore and incorporate in the future.

6. References

Dataset

1. F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872

Alternated Minimization

2. Recommendation system for e-commerce using ... - researchgate. (n.d.). Retrieved April 15, 2022, from https://www.researchgate.net/publication/349161176_Recommendation_System_for_E-commerce_Using_Alternating_Least_Squares_ALS_on_Apache_Spark
3. Over nonconvex sets - arxiv.org e-print archive. (n.d.). Retrieved April 15, 2022, from https://arxiv.org/pdf/1709.04451.pdf

Probabilistic Matrix Factorisation

4. Probabilistic matrix factorization - neurips. (n.d.). Retrieved April 15, 2022, from https://proceedings.neurips.cc/paper/2007/file/d7322ed717dedf1eb4e6e52a37ea7bcd-Paper.pdf

Memory Based approach

5. An Analysis of Memory Based Collaborative Filtering Recommender Systems with Improvement Proposals Retrieved April 15, 2022, from https://upcommons.upc.edu/bitstream/handle/2099.1/22602/102384.pdf%3Bjsessionid%3DCAA5DC3B04ECDC29BD96E8458D5B92E6?sequence%3D1

Hybrid Recommenders: Incorporating Metadata Awareness into Latent Factor Models

6. Hybrid recommenders: Incorporating metadata awareness into ... (n.d.). Retrieved April 15, 2022, from https://www.researchgate.net/publication/266655039_Hybrid_recommenders_Incorporating_metadata_awareness_into_latent_factor_models