

**AY 2021/2022, Semester 2**

**DSA4212: Assignment 2 Question 2**

Group 9

Lee Wei Qing (A0205666B)

Madeline Lim Chia Bing (A0205053W)

Mabel Lee Wei Ling (A0204397B)

Wu Weiye (A0200578H)

[**1 Introduction**](#_rfq7nh39brv) **1**

[**2 Matrix Factorisation with Genre Matrix**](#_84icdw736zu7) **1**

[2.1 Gradient Descent methods](#_3pvxybnd9s26) 1

[2.2 Broyden–Fletcher–Goldfarb–Shanno (BFGS)](#_f3uqk3cm3dty) 2

[**3 Low Rank Matrix Factorisation**](#_iz6t5vsda4aa) **3**

[3.1 Stochastic Gradient Descent with Momentum (SGDM)](#_851jxsmksr1d) 3

[3.2 Broyden–Fletcher–Goldfarb–Shanno (BFGS)](#_6lwrdmz8oewg) 3

[**4 Literature Review**](#_ue3h9fcjhjlh) **4**

[4.1 Alternating Least Squares (ALS)](#_eqiayi9oc8f) 4

[4.1.1 Alternating Least Squares with Weighted Regularisation (ALS-WR)](#_oq96ttjdbeqd) 4

[4.1.2 Implementation of ALS-WR](#_qmvvqgurb38) 5

[4.1.3 Alternating Least Squares (ALS)](#_hqpcmuj5e7z1) 6

[4.1.4 Implementation of ALS](#_rfwvsiejrnxz) 6

[4.2 Factorisation Machines (FM)](#_rhi665v60dq7) 7

[4.3 Memory Based Collaborative Filtering](#_taaa9i2q2bjj) 8

[4.3.1 Common Preference Degree](#_3exfqaynehd8) 8

[4.3.2 Rating Difference between Common Items](#_bkzi34dwr1xu) 9

[4.3.3 Collaborative Filtering Algorithm with Proposed Similarity Measure](#_gjo9ti81j76o) 9

[4.3.4 Implementation](#_eozawzmfv318) 10

[**5 Evaluation**](#_in686cove0v8) **10**

[**6 Conclusion**](#_vf6aj4y2ptqh) **11**

[**7 References**](#_twk8gkssgry0) **12**

## **1 Introduction**

In this project, we aim to predict the movie ratings made by each user. To solve this problem, we implemented baseline collaborative filtering methods with matrix factorisation and several optimisation strategies like Stochastic Gradient Descent with momentum (SGDM) and Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS). Furthermore, we employed the low rank approximation to compute matrix factorisation with lower dimension, in order to improve the efficiency of training and maximising the prediction performance of our model. Subsequently, we conducted literature review readings on alternative approaches to this movie ratings prediction problem and experimented with memory-based collaborative filtering methods and Alternating Least Squares (ALS). Following which, we compared the result of our models to literature reviews on existing complex models as part of the evaluation of our proposed baseline models.

Our dataset has three sections namely the users demographics, movies and ratings dataset. There are a total of 1,000,209 ratings with 6040 unique users who have made at least one rating in the ratings dataset and 3883 unique movies. We used the movies dataset to determine the set of genres classified under each movie. There are a total of 18 unique genres.

## **2 Matrix Factorisation with Genre Matrix**

We implemented Matrix Factorisation by initialising a 6040 by 18 user matrix with random numbers to represent the weights of the user, and a 18 by 3883 movie matrix obtained from one hot encoding the 18 genres of the movies into binary values. At every training step, the user matrix and movie matrix will be updated. The product of the user matrix and movie matrix would output a 6040 by 3883 matrix consisting of the predicted ratings made by each user on each movie. This matrix is indexed on the list of users and movies are taken from the ratings dataset, to ensure that computation of the Root Mean Square Error (RMSE) loss only takes into account the user-movie pair that has been rated.

### 2.1 Gradient Descent methods

To optimise the minimisation of training loss, we employed Stochastic Gradient Descent (SGD) and SGD with momentum (SGDM) to determine the best set of weights in the user matrix and in the movie matrix. Both SGD and SGDM are trained with a batch size of 5000 and learning rate of 0.9, and gamma of 0.9 in SGDM. The algorithms are trained based on the following formula below, to update both the weights of the user matrix and movie matrix in a parallel manner at every training step.

where and refers to the user matrix and the movie matrix respectively at each iteration, refers to the learning rate, and refers to the gradient of Loss function with respect to U and V.

It is not surprising that SGDM converges more efficiently within 30 epochs (refer to Fig. 1). This model achieved a train and test Root Mean Square Error (RMSE) of 0.785 and 0.857 respectively. We saw that the train error and the test error converge to similar MSE values (refer to Fig. 2), which implies that there is no overfitting in our fitted model. This is due to the large scale of the dataset and thus, no regularisation is required for this model.

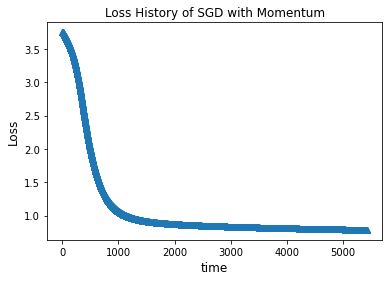
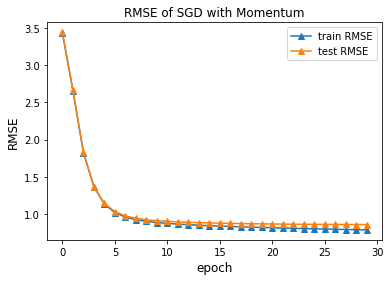
 

Fig. 1: Loss History of SGDM Fig. 2: Plot of Train and Test MSE

### 2.2 Broyden–Fletcher–Goldfarb–Shanno (BFGS)

The Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm is a second order optimisation algorithm belonging to the Quasi-Newton methods. It is commonly used for non-linear optimisation problems and approximates the second order derivative, the Hessian matrix, as opposed to the first order approximation method used in SGDM. We explored the BFGS algorithm for greater efficiency in locating the minimum point of this problem.

We first concatenate the weights for the user and movie matrix into a single long vector and trained the vector using the RMSE as loss metric. At each training step, we extract the weights of the user and movie matrix from the long vector and derive the train and test MSE. Using the spicy library, we trained our model on the L-BFGS-B method with 250 iterations and found that the fitted model converged quickly to train and test RMSE of 0.774 and 0.917 respectively (refer to Fig. 3).

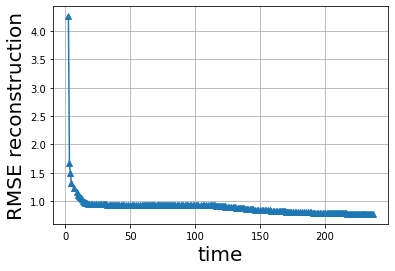


Fig. 3: Loss history of the model with BFGS algorithm

## **3 Low Rank Matrix Factorisation**

Low rank matrix factorisation is a method usually employed to approximate a large matrix with matrices of lower ranks. It allows us to find latent factors that are present in the matrices and deal with data sparsity which is evident in this dataset.

In an attempt to improve our current model, we experimented with low rank matrix factorisation to decompose our ratings matrix into low rank matrices. To determine the best rank that would maximise the prediction performance, we split our training data to obtain a validation set and experimented with different rank values to find the rank which performs best on the validation set. We fitted our model similarly to that in Section 2. However, instead of using the movie matrix (18 by 3883) containing information on the 18 genres of the movies available, which has a rank of 18, our model is trained on a movie matrix with lower rank and initialised with 2 randomised matrices.

### 3.1 Stochastic Gradient Descent with momentum (SGDM)

Using SGDM as an optimisation method, we trained our model across a range of ranks from 1 to 50. We noted that the test RMSE increases while train RMSE continues to decrease when larger rank values are used which occurs due to overfitting. This is not surprising as the model has probably overfitted with too many latent variables. To narrow down our selection of best rank, we zoomed in to ranks 1 to 9. As seen in the plot on the right, using a rank of 3 is optimal and returns the best performance with a train RMSE of 0.883, and validation RMSE of 0.908 (refer to Fig. 4).

Subsequently, using the model trained on 90% of the available ratings with rank of 3, we obtained a test RMSE of 0.901 on the remaining 10% of the data.

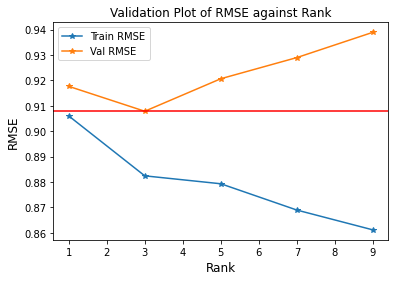
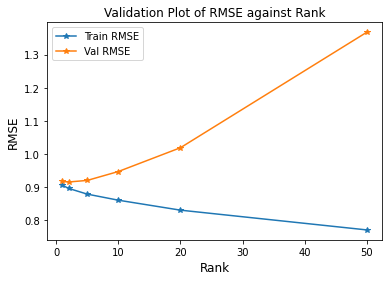


Fig 4: Validation Plot for Rank size using SGDM

### 3.2 Broyden–Fletcher–Goldfarb–Shanno (BFGS)

Using a validation set, we trained our model using the BFGS optimisation algorithm across ranks from 1 to 20. The optimal rank derived is 7, which gives a predicted validation RMSE of 0.867 (refer to Fig. 5). Choosing a rank with value higher than 7 would result in overfitting and rise in validation RMSE. Using this finding, we trained our model by initialising the user matrix and movie matrix with rank of 7 and ran 250 iterations. This model achieved a train and test RMSE of 0.774 and 0.915.

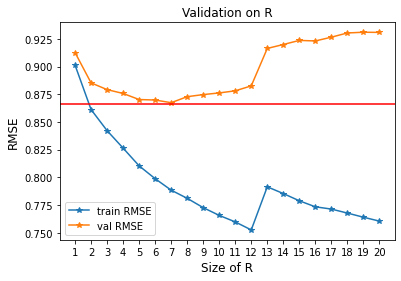


Fig. 5: Validation Plot for Rank size using L-BFGS

## **4 Literature Review**

To evaluate the performance of our baseline models in Section 2 and 3, we conducted literature reviews on collaborative filtering on existing complex models. We were looking out for similar models that we could possibly try and evaluate against our own, as well as other diverse and novel approaches to understand the field and how other researchers are addressing the problems we are facing. Wherever possible, we implemented the models we read up on, so that we could hold the results against our own and see how it performs on the same dataset.

### 4.1 Alternating Least Squares (ALS)

Alternating Least Squares is one of the approaches to perform low-rank matrix factorisation, and it is famously known for its quick linear time convergence.

In the low-rank matrix factorisation we tried, the cost function is non-convex which makes it difficult as an optimisation problem. Hence, ALS attempts to ease the process by treating it as two different convex optimisation problems. We first fixed the users matrix() and minimised the convex function of movies matrix before fixing the movies matrix() and optimising that of the users matrix, resulting in an algorithm that converges within linear time.

#### 4.1.1 Alternating Least Squares with Weighted Regularisation (ALS-WR)

In ALS, there are many free parameters, specifically (|| + ||) x r of them, where || represents the cardinality of the matrix, and r is the rank we choose. This, accompanied with the reality of a sparse dataset, means that ALS would be likely to overfit the data more often than not.

In the paper written by Zhou. et al [2] who were competing for the 2008 Netflix prize to improve RMSE by 10%, they explored a novel method of ALS-WR, in attempt to address the overfitting problem by adding a weighted regularisation term, and they found that the data never overfits even as they increase the number of features(r) or number of iterations. The target function we are trying to optimise is as follows:

*ij - ij )2* ui||i||2 *vj*||*j*||2 )

where is the users matrix, is the movies matrix, and  is the ratings matrix. We let be the set of movies that user has rated, and be the cardinality of it. Likewise, we let be the set of users that rated movie *,* and be the cardinality of it.

#### 4.1.2 Implementation of ALS-WR

We then implemented ALS-WR by recursively updating and using the formulas:

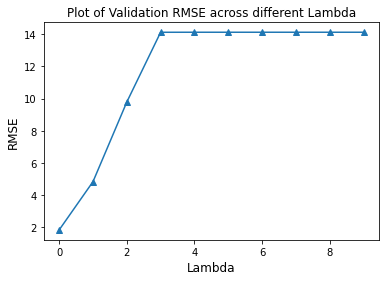
*i  i-1* *Li*

where *i* ɪ *, Li* and denotes the sub-matrix of where columns ∈ are selected, and is the row vector where ∈ i of the -th row of is taken.

*j  j-1* *Lj*

where *i* ɪ *, Li* and denotes the sub-matrix of where rows ∈ are selected, and is the column vector where rows ∈ of the -th column of is taken.

Theoretically, we expected the model to not overfit as mentioned by Zhou. et al [2], with validation MSE to remain stable with increasing ranks and iterations. However, when we did validation for different values of rank(r) and *,* validation RMSE remained constant for the same lambda despite increasing rank, showing that ALS-WR is not sensitive to rank values (refer to Fig. 6). From the plot below, we see that RMSE increases steadily from 0 when increases, which is odd as it would mean that optimising the loss function alone is enough to prevent overfitting on our training set.

  
Fig. 6: Plot of RMSE against Lambda

#### 4.1.3 Alternating Least Squares (ALS)

We then turned to the original ALS algorithm to see how it would hold up against the weighted regularised version we read on and tried above. For ALS, the function we are trying to optimise is as follows, which has the cardinality of the rows and columns removed from the regularisation term:

*ij - ij )2* ||i||2 ||*j*||2 )

The recursive updating follows the below closed form equation:

*-1*

*-1*

where is the target rating and is the regularisation weight.

#### 4.1.4 Implementation of ALS

Using a validation set, we obtained the optimal rank of 26 and where validation RMSE is the lowest, as seen in the plot in (refer to Fig. 7 and Fig. 8 left) . We then trained the model with rank of 26 with the ALS algorithm and achieved the train and test RMSE of 2.5662 and 2.7021 respectively.

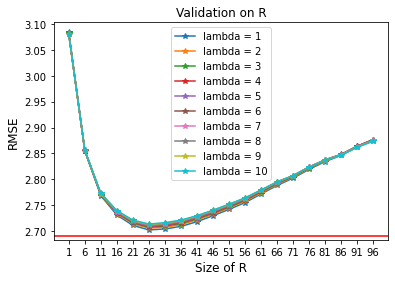


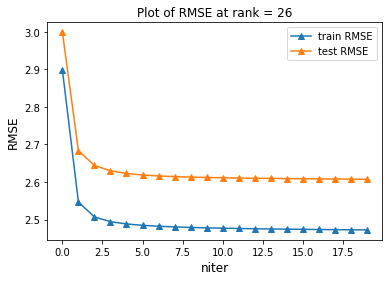
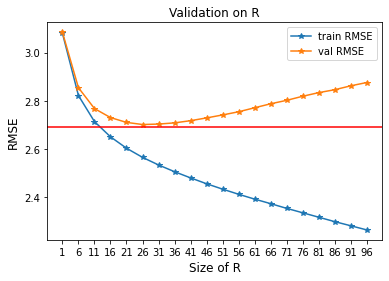
Fig. 7: Plot of validation RMSE against rank R for different values  


Fig. 8: Validation Plot (left) and Plot of train and test RMSE, R = 26, =1 (Right)

Indeed, looking from the plots on the right (refer to Fig. 8), we see that ALS converges within very few iterations, and the whole operation takes less than a minute on such a huge dataset. This is an extremely efficient operation that our other models cannot compare with. However, there really is no free lunch, as the accuracy suffers even with the best parameters selected. We hypothesise that the naive assumptions made by ALS on the convexity of the data makes it hard for ALS to approach the truth, therefore we saw how the RMSE was comparatively higher than other models.

One strong trait of ALS is its ability to deal with cold-start problems though, as the matrices generated when multiplied together, is able to make predictions for movies that have never been rated in the dataset. Therefore, while ALS generally performs poorly, we can possibly think of how we can tap on this trait and use ALS in conjunction with better-performing models.

While it would have been rewarding to be able to investigate and understand how to use this group of ALS models to efficiently harness the power of a large dataset, regrettably we do not have the mathematical expertise to delve deeper into this model.

### 4.2 Factorisation Machines (FM)

Another interesting alternative approach to this prediction problem would be to implement Factorisation Machines (FM) to improve the efficiency of the training, while taking into account other features interaction which includes the user demographics into the fitted model.

In the paper by Rendle [3], he introduced FM which is a generic supervised learning method that is built on Support Vector Machine (SVM) with factorised parameterisation instead of the dense parametrisation in conventional SVM. A typical matrix factorisation trains with the product of the matrices directly related to the weights representing the user and the films to get a matrix with the size of number of users by number of films. However, FM is able to deal with sparse data which are also augmented with arbitrary auxiliary features (e.g the user demographics, movie duration) and map the interactions between variables into a low dimensional latent factor (refer to Fig. 9).

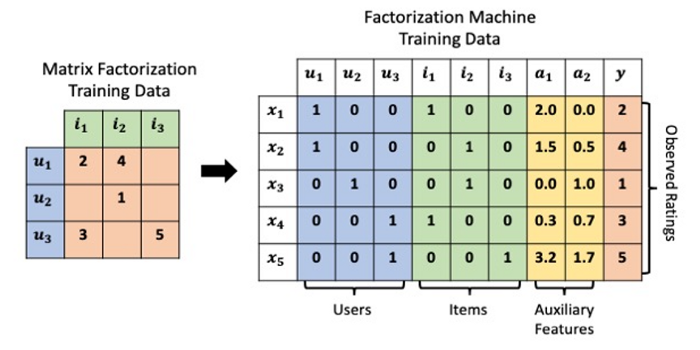


Fig. 9: Picturing the difference between Matrix Factorisation and Factorisation Machine [4]

Using the following FM equation, the user-movie pairwise interactions could be approximated and only linear time is required for such a computation.

where is the global bias (or intercept), is the strength of the i-th variable and is the interaction between the i-th and j=th variable.

Although the FM model would be an efficient algorithm to explore for this problem, we acknowledged that implementing and optimising this FM model would be highly complex and beyond our capacity and time constraint. This is due to the nature of its equation which requires strong mathematical computation. Furthermore, there are only a splattering number of works done using this model which increases the difficulty in finding resources and guides for this implementation. Further work could be done to employ this model for our problem.

### 4.3 Memory Based Collaborative Filtering

We have been exploring various model based collaborative filtering methods which aim to learn a set of weights from the training set and use it for prediction. Another collaborative filtering method that is widely used in the industry is the memory based collaborative filtering, which generates the predicted ratings through the measurement of the similarity between users or items. The predicted ratings are computed based on the weighted average of the mean of the active user's neighbour and addition of the active user’s mean.

In the paper by Zhang. et al [1], they highlighted some of the problems with utilising the traditional similarity measurements such as cosine similarity and Pearson correlation on sparse data: expensive computations, large memory usage and lower accuracy. The researcher proposed a modified similarity measurement to improve its accuracy especially for sparse data.

#### 4.3.1 Common Preference Degree

Traditional similarity measurements do not take into account the number of common items between the 2 users. This would result in unusually high similarity scores between the 2 users even though they have little common rated items. For example, suppose a situation where users A and B who have only watched 1 movie in common give the same rating to the movie. Whereas on the other hand, users A and C who have watched multiple movies in common give differing ratings to the movies they have watched. Since users A and C watched a similar set of movies, we would think that users A and C are quite similar. However, traditional similarity measurements will classify user A to be much more similar to user B instead of user C. As such, the researchers proposed to include a term for common preference degree in the similarity metric.

(1)

(2)

where represents the common preference degree between user a and user b. It is calculated by taking the ratio of the number of commonly rated items between user a and user b over the maximum number of common rated items between user a and all other users as shown in equation (2).

#### 4.3.2 Rating Difference between Common Items

The researcher also proposed adding the rating difference between the common rated items.

(3)

(4)

(5)

where di represents the difference between user a and b for item i. The rating difference between user a and b can be calculated as shown above in equation (4). The similarity score is then defined as shown in equation (5), where rmax and rmin represent the maximum and minimum values of the two user ratings on the common rated item set.

#### 4.3.3 Collaborative Filtering Algorithm with Proposed Similarity Measure

The final similarity metric is defined as follows, where is a parameter to be chosen:

(6)

Using the final similarity metric above, we find the the 5 nearest neighbours (denoted by NBS) for each active user a and predicted using the formula as follows:

(7)

where f refers to the film and refers to the average film ratings of the user a.

#### 4.3.4 Implementation

We first used cosine similarity, one of the traditional similarity measurements, to compute the similarity between the users using the 90% training data. We then computed as shown in equation (7) for the 10% test data. The resulting model reported a RMSE of 1.010. We then replace cosine similarity with the improved similarity metric as shown in equation (6). Using cross validation, we observed that the optimal is 0.5 (refer to Fig. 10). The resulting model reported a high performance with RMSE of 0.611.

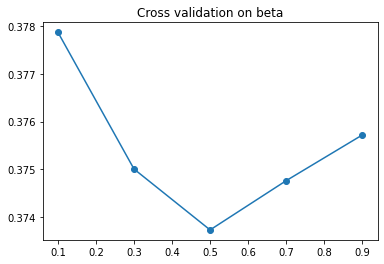


Fig. 10: Cross validation on

## **5 Evaluation**

In this section, we compared the various methods and optimisation approaches we have experimented with (refer to Table 1).

We implemented our baseline models using the matrix factorisation and low rank matrix factorisation approach. We chose these two as we wanted models that were robust and known to work well, yet simple to train, such that we can hold other more complex models that we decide to branch into against these benchmarks. These will ensure that we are not unnecessarily implementing more complex models which end up performing the same as our baseline models.

For each approach, we explored two optimisation algorithms, the SGDM and L-BFGS, independently to minimise the loss function. While L-BFGS is comparable at best in terms of its prediction accuracy compared to SGDM, it outstrips SGDM significantly in terms of training efficiency, as this Quasi-Newton method iterates much faster than SGDM.

Generally comparing between the 2 models, MF with Genre Matrix and Low Rank MF, we found that the first model, which initialises the movie matrix with binary values to indicate genre, performs better than the latter Low Rank MF, which initialises with random weights.

We postulated that this could be due to importance in the role of genre in categorising each movie, which resulted in the underlying relationship being captured as we trained the model. Moreover, there could also be some association between the genres and each user due to user preferences.

With these 4 models as our baseline, the test MSE of about 0.9 is now the benchmark to reach.

| **Model** | | **Test RMSE** |
| --- | --- | --- |
| Matrix factorisation(MF) with genre matrix | SGDM | 0.857 |
| L-BFGS | 0.917 |
| Low rank matrix factorisation | SGDM | 0.901 |
| L-BFGS | 0.915 |
| ALS | | 2.607 |
| Cosine Similarity | | 1.010 |
| Improved Cosine Similarity (Fsim) | | 0.611 |

Table 1: Reports the Test RMSE for each approach and optimisation methods

Now moving onto the models we implemented based on literature reviews, we see that ALS-WR performed significantly worse than all our other models, likely due to the naive assumptions about the convexity made. That said, ALS precedes all other models in terms of training efficiency, as the convergence of the matrix happens within a minute despite the large size.

Finally, we explored memory based collaborative filtering methods which focus on similarity between users to predict ratings. While using the traditional similarity metric of cosine, the model reported a test RMSE of 1.010 which does not seem ideal. Upon implementing the improved cosine similarity (Fsim), the model reported a test RMSE of 0.611 which performs better than the traditional similarity metric and also, other model based methods. However, it is crucial to note that memory based methods might not be scalable with larger datasets, which is a common occurrence in the real world. In instances where scalability is concerned, a model based collaborative filtering method will be chosen instead.

## **6 Conclusion**

In conclusion, our group utilised two main approaches to build baseline models optimised by SGDM and L-BFGS algorithms for this movie rating prediction problem: firstly, the matrix factorisation using genre matrix and secondly, the low rank matrix factorisation. It was found that the low rank matrix factorisation approach had marginal improvement as compared to the matrix factorisation with genre matrix approach. It is also reported that although SGDM takes a longer training time as compared to L-BFGS, it converges to a lower test RMSE of 0.857 using the first approach.

With the application of literature review learnings, we implemented the memory based collaborative filtering model which uses an improved similarity metric and prediction function proposed by Zhang.

Our baseline model performs considerably well given the set of parameters we have tuned and chosen. However, if we were to choose a model that predicts the best, the Improved Cosine Similarity (Fsim) model is the clear winner, with a remarkable test RMSE of 0.611, a clear cut above the rest.

From this project, we learned that reading widely and learning from other people’s research is critical and edifying, as they inform us of pitfalls we might not not have considered and teach us of methods proven to work. The improved cosine similarity algorithm is one shining example of that, as we see how the vanilla cosine similarity optimization is adapted and improved upon to cover for its loopholes, and is able to get ahead of all other models in terms of prediction accuracy.  
  
That being said, we also acknowledge that not all fancy models will do well. Our baseline models, with the right hyperparameter tuning, were able to outperform most models and give us robust predictions. A wise approach for future projects would be to choose robust models, and conduct literature reviews on them first before reading up on more complex, novel approaches.

As part of our future work, we could further explore the models that we mentioned would take more understanding to implement, like Alternating Least Squares (ALS) and Factorisation Machines (FM) methods. It would be extremely rewarding to be able to see results parallel to those described in the literature that we reviewed, and to appreciate what these authors have poured their heart into.

## **7 References**

[1] Zhang Baofu, et al (2017, May 8) Improved collaborative filtering recommendation algorithm of similarity measure. Retrieved April 12, 2022, from <https://aip.scitation.org/doi/pdf/10.1063/1.4982532>

[2] Zhou, Y., Wilkinson, D., Schreiber, R., &amp; Pan, R. (1970, January 1). Large-scale parallel collaborative filtering for the netflix prize. SpringerLink. Retrieved April 14, 2022, from <https://link.springer.com/chapter/10.1007/978-3-540-68880-8_32>

[3] Rendle, S. (2010). Factorization machines - 國立臺灣大學. Retrieved April 12, 2022, from <https://www.csie.ntu.edu.tw/~b97053/paper/Rendle2010FM.pdf>

[4] Lundquist, E. (2020, June 29). Factorization machines for item recommendation with implicit feedback data. Medium. Retrieved April 12, 2022, from <https://towardsdatascience.com/factorization-machines-for-item-recommendation-with-implicit-feedback-data-5655a7c749db>