Collaborative Filtering

University of California, Los Angeles
ECE 219 Project 3

Joshua Hannan (UID: 805221851)
James (Zach) Harris (UID: 105221897)
Dennis Wang (UID: 105229662)
Akash Deep Singh (UID: 405228294)

jhannan@g.ucla.edu
jzharris@g.ucla.edu
dmwang626@g.ucla.edu
akashdeepsingh@g.ucla.edu

1 Introduction

In this project, we aim to build a recommender system to infer user interests based on data from that user and other users with "similar" interests. To do so, we will be using collaborative filtering models in which we analyze the relationship between users and items. Specifically, we will be creating a system that can recommend movies to users based on other user ratings.

2 Collaborative Filtering Models

Collaborative filtering models look at already-existing user-item relationships to extrapolate how a specific user will react to an item he/she has not already seen. This technique is particularly useful in sparse data sets, where users have not seen many items. For our data set, there are many users that have rated only a few movies, and therefore our data set will be very sparse. However, with various collaborative filtering techniques, namely neighborhood-based and model-based collaborative filtering, we will create a recommender system that works well on this sparse data set.

3 MovieLens Dataset

We will be using the small MovieLens data set, which has approximately 100,000 ratings from 600 users and over 9000 movies. As we will see, this is a very sparse data set. We are also given information about the movie titles and genres, so we can interpret how well our recommender system performs based on this.

Question 1 We found the sparsity to be

$$\text{Sparsity} = 0.016999683055613623$$

based on the formula: $\text{Sparsity} = \frac{\text{Total number of available ratings}}{\text{Total number of possible ratings}}$.

Question 2 Figure 1 shows the frequency of rating values. We can see that the most popular ratings are 3.0 and 4.0, and that the distribution is skewed left. There are very few ratings less than 2.0, showing a bias in people's ratings.

Question 3 Figure 2 shows the distribution of number of ratings among movies. We see that the most watched movie has over 300 ratings, while most other movies have close to 0 ratings.

Question 4 Figure 3 shows the distribution of number of ratings among users. The most ratings a single user has given is over 2500, while the vast majority have given fewer than 500.

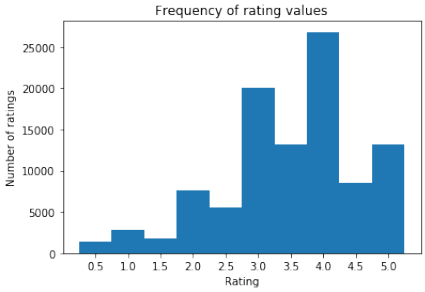

Fig. 1. Frequency of rating values

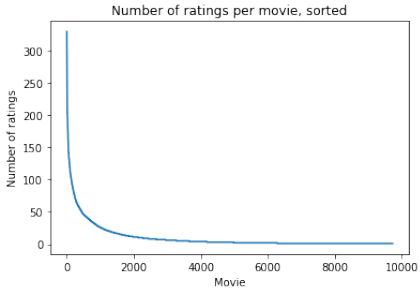

Fig. 2. Distribution of ratings among movies

Question 5 The distribution of number of ratings among movies is heavily skewed to the right, which means that very few movies have a large number of ratings. This means that our R matrix of ratings will be extremely sparse, so our recommendation system will not have too much information. This can lead to poor recommendations, as our collaborative filtering models perform better with more data.

Question 6 Figure 4 shows the variance in rating values. We notice that the vast majority of movies have variance $\sigma^2 \leq 1$. This can be attributed to the fact that movies are usually consistently rated among most users, and very few movies have a wide spread of ratings.

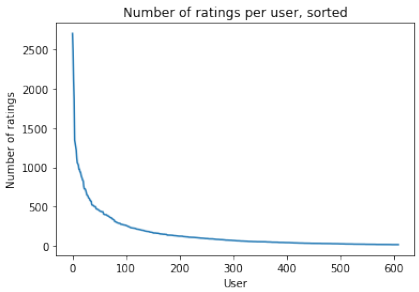

Fig. 3. Distribution of ratings among users

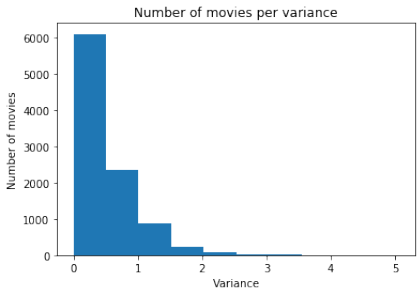

Fig. 4. Variance of ratings among movies

4 User-based Collaborative Filtering

4.1 User-based Neighborhood Models

User-based neighborhoods are employed to identify similar users to the target user for whom the rating predictions are being computed. To determine the

neighborhood of the target user u, her similarity to all the other users is computed. Therefore, a similarity function is defined between the ratings specified by users.

4.2 Pearson-correlation coefficient

We use Pearson-correlation coefficient to compute the similarity between users. Pearson-correlation coefficient between users u and v, denoted by Pearson(u, v), explains the similarity between the rating vectors of users u and v.

Question 7 The mean rating for user u computed using their specified ratings, μ_u, can be written in terms of the set of item indices for which ratings have been specified by user u, I_u, and the rating of user u for item k, r_{uk}, as the following:

$$\mu_u = \sum_{k \in I_U} \frac{r_{uk}}{|I_u|} \quad \forall u \in \{1 \dots m\}.$$

Question 8 The **indices of movies that have been rated by both user u and user v can be represented as $I_u \cap I_v$.** $I_u \cap I_v = \emptyset$ represents when user u and user v have not rated any movies in common. **It is possible that $I_u \cap I_v = \emptyset$ when the rating matrix R is sparse.**

The Pearson-correlation coefficient between users u and v is defined by:

$$Pearson(u, v) = \frac{\sum\limits_{k \in I_u \cap I_v} (r_{uk} - \mu_u)(r_{vk} - \mu_v)}{\sqrt{\sum\limits_{k \in I_u \cap I_v} (r_{uk} - \mu_u)^2} \sqrt{\sum\limits_{k \in I_u \cap I_v} (r_{vk} - \mu_v)^2}}$$

4.3 k-Nearest Neighborhood

The k-Nearest Neighbor function of user u, given by P_u, is the set of k users with the highest Pearson-correlation coefficient with user u.

4.4 Prediction Function

The prediction function for user-based neighborhood model is given by

$$\hat{r}_{uj} = \mu_u + \frac{\sum\limits_{v \in P_u} Pearson(u, v)(r_{vj} - \mu_v)}{\sum\limits_{v \in P_u} |Pearson(u, v)|}$$

Question 9 Mean-centering the raw ratings $(r_{uk} - u_v)$ in the prediction function is **implemented to mitigate discrepancies in the scales that users use to provide ratings.** For example, one user might rate all items highly, while another user might rate all items poorly. Consequently, mean-centering is employed to enable a better "relative" prediction with respect to ratings that have been observed.

4.5 k-NN Collaborative Filter

Question 10 Figures 5 and 6 Show the average RMSE and MAE respectively against the number of latent factors used in the NMF algorithm.

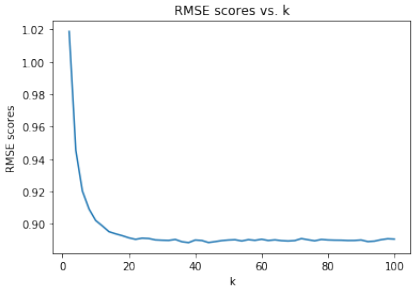

Fig. 5. RMSE, k-NN, Full Test Set

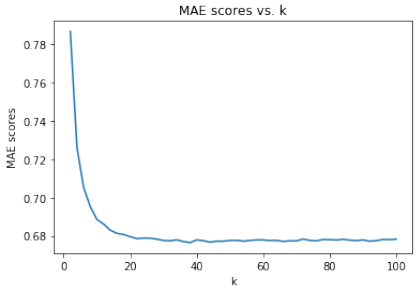

Fig. 6. MAE, k-NN, Full Test Set

Question 11 From fig 5, we notice that the value of 'minimum k' is close to 20 as it is the value after which the RMSE and MAE curves converge to a steady-state value. Just to be on the safe side, we select the value of $k = 22$ as the curves stabilize after this value.

Steady State values of:

- RMSE = 0.8905
- MAE = 0.6785

4.6 Filter Performance on Trimmed Dataset

For this part, we filter out part of the test set using various trimming techniques. First, we perform popular movie trimming in which only the movies that have received more than 2 ratings are kept. The unpopular movie test set contains only movies that have 2 or fewer ratings. Finally, the high variance movie test set only contains movies that have at least 5 ratings and a variance of at least 2. These different trimming methods allow us to see how well our recommender system performs in various situations, such as data sets that have many data points (popular), those that are very sparse (unpopular), and those that are all over the place in their data points (high variance).

Question 12 Figure 7 shows the average RMSE scores on the popular movietrimmed test set. We see that the optimal number of latent factors is, $k = 46$, which gives us an average RMSE of 0.8535.

Question 13 Figure 8 shows the average RMSE scores on the unpopular movie trimmed test set. We see that the optimal number of latent factors is $k = 86$, which gives us an average RMSE of 0.9528.

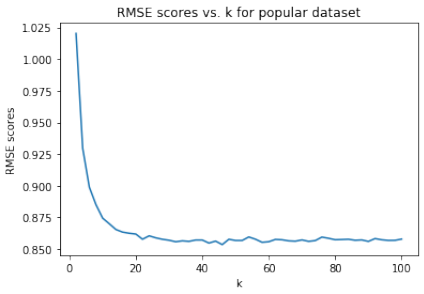

Fig. 7. RMSE, k-NN, Popular Test Set

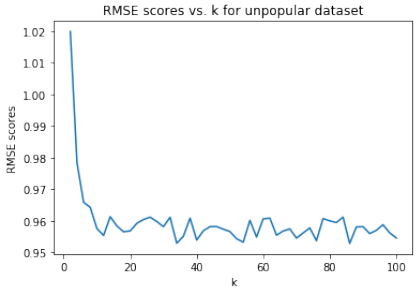

Fig. 8. RMSE, k-NN, Unpopular Test Set

Question 14 Figure 9 shows the average RMSE scores on the unpopular movie trimmed test set. We see that the optimal number of latent factors is $k = 20$, which gives us an average RMSE of 1.3732.

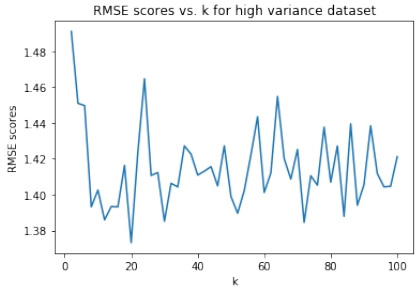

Fig. 9. RMSE, k-NN, High Variance Test Set

Question 15 Figures 10-13 show the ROC curves, along with the areas under the curve. The optimal $k = 21$ was used. The area under the curve for each ROC curve is reported on the plot itself.

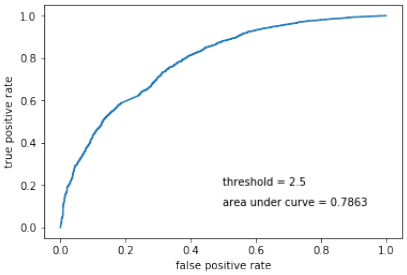

Fig. 10. ROC, k-NN, Threshold = 2.5

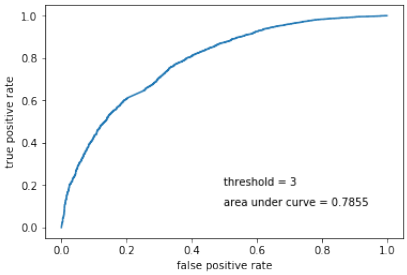

Fig. 11. ROC, k-NN, Threshold = 3.0

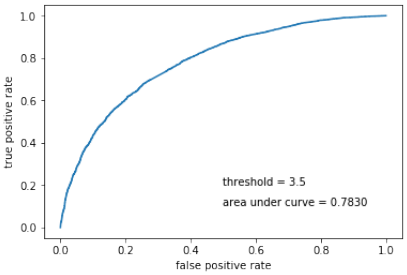

Fig. 12. ROC, k-NN, Threshold = 3.5

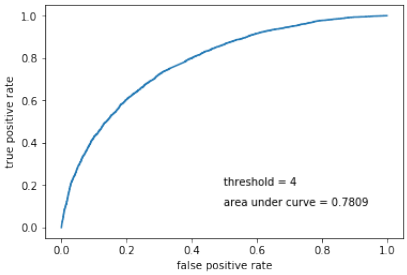

Fig. 13. ROC, k-NN, Threshold = 4.0

5 Model-based Collaborative Filtering

5.1 Latent Factor Based Collaborative Filtering

In general, latent factor-based collaborative filtering is done using matrix factorizations to attempt to fill in the missing values in our rating matrix. The general form for this optimization problem is:

$$\underset{U,V}{\text{minimize}} \sum_{i=1}^{m}\sum_{j=1}^{n} W_{ij}(r_{ij} - (UV^T)_{ij})^2 \tag{1}$$

If we want to have regularization, we add two terms $\lambda\|U\|_F^2 + \lambda\|V\|_F^2$, where λ is a hyperparameter that we select.

5.2 Nonnegative Matrix Factorization

Nonnegative matrix factorization, or NMF, is a particularly useful matrix factorization technique for rating systems because NMF approximates the original

ratings matrix with matrices that are elementwise nonnegative. This gives a high sense of interpretability of these factors. This problem has the form:

$$\underset{U,V}{\text{minimize}} \sum_{i=1}^{m}\sum_{j=1}^{n} W_{ij}(r_{ij} - (UV^T)_{ij})^2 \quad \text{s.t. } U_{ij}, V_{ij} \geq 0$$

Question 16 Equation 5.1 is not inherently convex, due to the nonlinearity introduced when optimizing UV^T. However, alternating least-squares (ALS) can solve this problem by fixing one of these variables and solving for the other. In this way, we can fix U and turn Equation 5.1 into the following convex least-squares problem:

$$\underset{V}{\text{minimize}} \sum_{i=1}^{m}\sum_{j=1}^{n} W_{ij}(r_{ij} - (UV^T)_{ij})^2$$

We know that this is a convex problem, because $r_{ij} - (UV^T)_{ij}$ is affine since U is now a scalar, and the square of an affine function is convex.

Question 17 Figures 14 and 15 Show the average RMSE and MAE respectively against the number of latent factors used in the NMF algorithm.

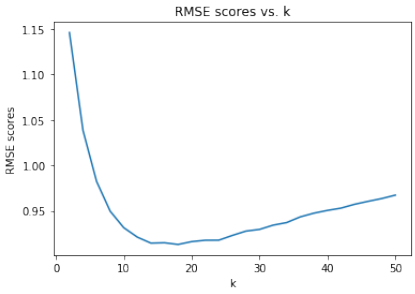

Fig. 14. RMSE, NMF, Full Test Set

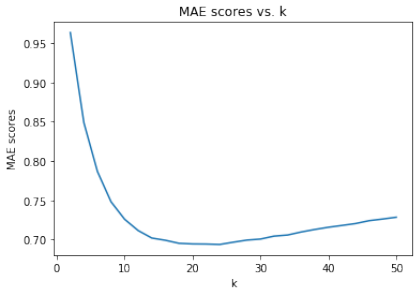

Fig. 15. MAE, NMF, Full Test Set

Question 18 From figures 14 and 15, we see that the optimal number of latent factors is 18 and 24 for RMSE and MAE. These give errors

$$\text{minimum average RMSE} = 0.9130$$
$$\text{minimum average MAE} = 0.6939$$

There are 18 genres of movies in the Movie Lens dataset, which is precisely the optimal number of latent factors for RMSE. While the optimal k for MAE is higher at 24, we can see from the plots that $k = 18$ achieves a very similar MAE. Thus the optimal number of latent factors is essentially the same as the number of movie genres.

Question 19 Figure 16 shows the average RMSE scores on the popular movie trimmed test set. We see that the optimal number of latent factors is $k = 16$, which gives us an average RMSE of 0.8709.

Question 20 Figure 17 shows the average RMSE scores on the unpopular movie trimmed test set. We see that the optimal number of latent factors is $k = 22$, which gives us an average RMSE of 0.9907.

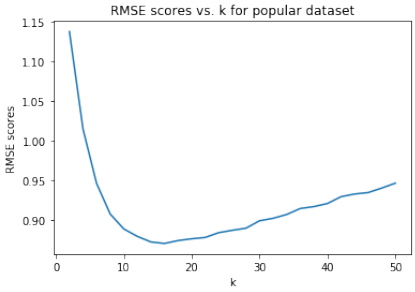

Fig. 16. RMSE, NMF, Popular Test Set

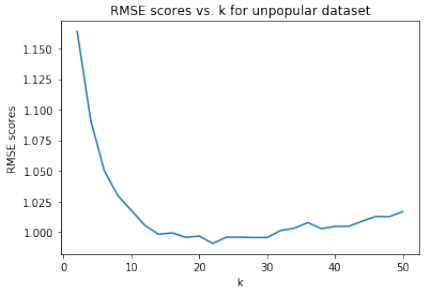

Fig. 17. RMSE, NMF, Unpopular Test Set

Question 21 Figure 18 shows the average RMSE scores on the high variance movie trimmed test set. We see that the optimal number of latent factors is $k = 6$, which gives us an average RMSE of 1.382.

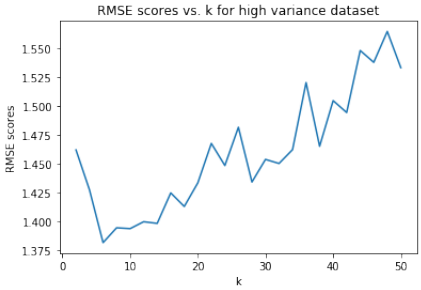

Fig. 18. RMSE, NMF, High Variance Test Set

Question 22 Figures 19-22 show the ROC curves, along with the areas under the curve. The optimal $k = 18$ was used.

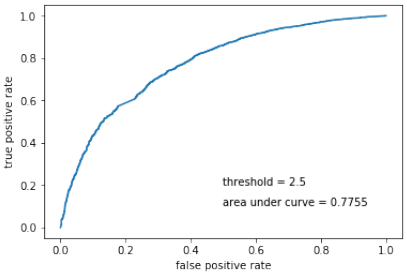

Fig. 19. ROC, NMF, Threshold = 2.5

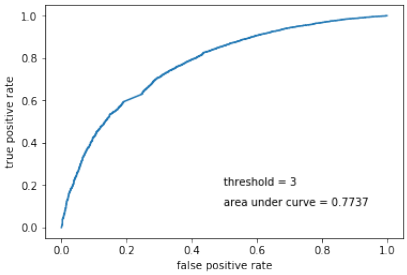

Fig. 20. ROC, NMF, Threshold = 3.0

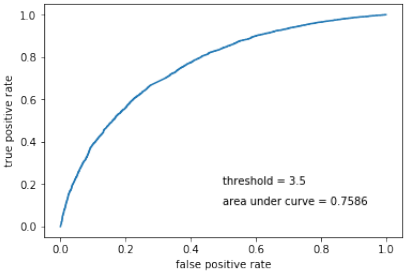

Fig. 21. ROC, NMF, Threshold = 3.5

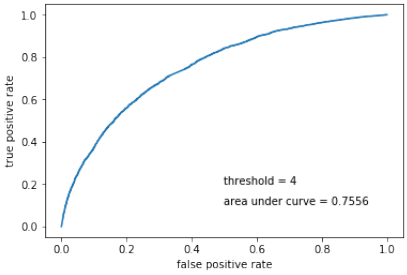

Fig. 22. ROC, NMF, Threshold = 4.0

Question 23 Table 1 shows the top 10 movies for the first 3 features. We notice that most of these top 10 movies belong to the genres "Drama" or "Comedy." This makes sense because if a person enjoys a specific genre of movie, say "Drama," then our recommendation system should tend to recommend more movies that are similar, namely in the same genre. However, this bias towards just a few genres could be from the possibility that there are simply more movies of those genres in the data set.

We notice that the first feature has a lot of movies from the "Drama" genre, among some other genres. Feature 2, while still having a lot of drama films, also contains multiple "Thriller" and "Comedy," and the 3rd feature has more "Children" and "Romance" movies. **From this trend, we can notice that each latent factor corresponds to a different subset of genres. Thus, there is a connection between the latent factors and the movie genres.**

5.3 Matrix Factorization with Bias

Matrix factorization (MF) with bias is similar to the previously used nonnegative matrix factorization with the addition of a bias term for each user and item. This

Feature 1	Feature 2	Feature 3
Comedy|Drama|Romance	Drama|Western	Comedy|Drama|Romance
Horror|Sci-Fi	Drama|Musical	Horror|Thriller
Crime|Drama|Musical	Film-Noir|Thriller	Drama
Comedy|Drama|Romance	Comedy	Drama|Romance
Drama|Romance	Comedy|Romance	Adventure|Children
Comedy	Comedy	Drama|Romance
Drama|Mystery|Sci-Fi|Thriller	Action|Comedy	Drama
Drama	Action|Crime|Thriller	Crime|Drama|Film-Noir|Thriller
Drama	Horror|Sci-Fi|Thriller	Drama
Drama|Thriller|War	Children|Comedy	Adventure|Children|Comedy|Mystery

Table 1. Top 10 movies per feature

modified optimization equation ahs the following form:

$$\underset{U,V,b_u,b_i}{\text{minimize}} \sum_{i=1}^{m}\sum_{j=1}^{n} W_{ij}(r_{ij}-\hat{r}_{ij})^2 + \lambda\|U\|_F^2 + \lambda\|V\|_F^2 + \lambda\sum_{u=1}^{m} b_u^2 + \lambda\sum_{i=1}^{n} b_i^2$$

where b_u is the bias of user u and b_i is the bias of item i. The optimization variables are U, V, b_u, b_i. The introduction of bias terms helps to remove the bias that certain users exhibit, or the bias for certain movies. For example, if a user rates movies two stars higher than other users, the introduction of a bias "penalization" will remove two stars from all of this user's ratings to compare their ratings to others'.

After solving for the optimization variables U, V, b_u, b_i, we predict the rating, $\hat{r}_{ij}$, for user i for item j using the following equation:

$$\hat{r}_{ij} = \mu + b_i + b_j + \sum_{s=1}^{k} u_{is} \cdot v_{js}$$

where μ is the mean of all ratings, b_i is the bias of user i, and b_j is the bias of item j.

For questions 24 to 29, we implement a MF with bias collaborative filter using the aforementioned techniques and 10-fold validation. We sweep the number of latent factors, k, from 2 to 50 in step sizes of 2, and for each k compute the average RMSE and average MAE across all 10 folds.

Question 24 Figure 23 is a plot of the average RMSE (Y-axis) against k (X-axis) and figure 24 is a plot of the average MAE (Y-axis) against k (X-axis) for the complete MovieLens dataset.

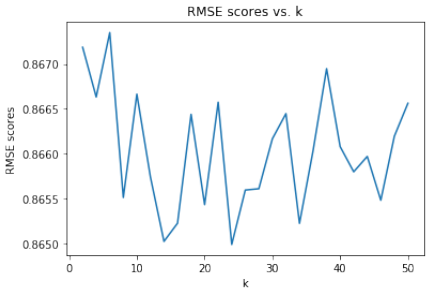

Fig. 23. RMSE, MF with Bias, Full Test Set

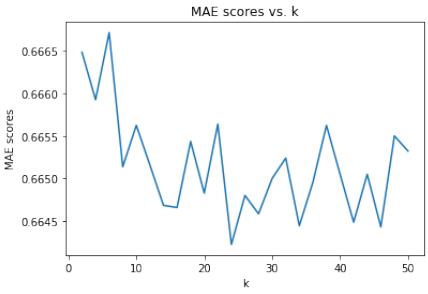

Fig. 24. MAE, MF with Bias, Full Test Set

Question 25 The optimal number of latent factors is the value of k that gives the minimum average RMSE or the minimum average MAE. The optimal k for MF with Bias using the full dataset that gives minimum average RMSE is $k = 24$. The optimal k that gives minimum average MAE is $k = 24$. Analyzing the graphs reveals that in general, **the optimal k value is around 24 for both the minimum average RMSE and MAE.** We can observe this by noticing the general convexity of both graphs, which have a minimum at $k = 24$.

$$\text{minimum average RMSE} = 0.8650320745151244$$
$$\text{minimum average MAE} = 0.6640776876193643$$

Question 26 Figure 25 is a plot of the average RMSE (Y-axis) against k (X-axis) using the popular movie trimmed test set.

The minimum average RMSE value is 0.8452120559192975.

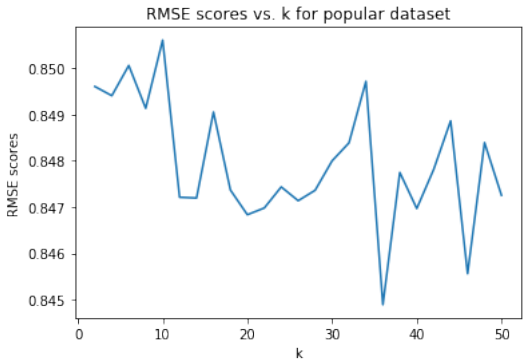

Fig. 25. RMSE, MF with Bias, Popular Movie Trimmed Test Set

Question 27 Figure 26 is a plot of the average RMSE (Y-axis) against k (X-axis) using the unpopular movie trimmed test set.

The minimum average RMSE value is 0.899150963345269.

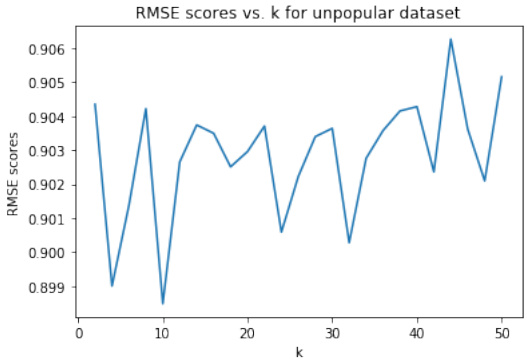

Fig. 26. RMSE, MF with Bias, Unpopular Movie Trimmed Test Set

Question 28 Figure 27 is a plot of the average RMSE (Y-axis) against k (X-axis) using the high variance movie trimmed test set.

The minimum average RMSE value is 1.3517553880939661.

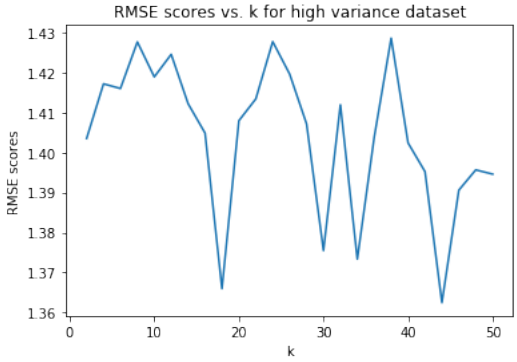

Fig. 27. RMSE, MF with Bias, High Variance Movie Trimmed Test Set

Question 29 Figure 28 is the ROC curve for MF with bias for threshold value 2.5. Figure 29 is the ROC curve for MF with bias for threshold value 3. Figure

30 is the ROC curve for MF with bias for threshold value 3.5. Figure 31 is the ROC curve for MF with bias for threshold value 4. All plots for optimal number of latent factors (26). Table 2 states the AUC values for each ROC plot with the aforementioned threshold values.

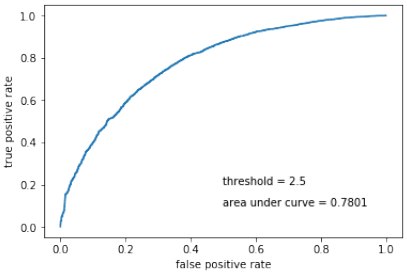

Fig. 28. RMSE, MF with Bias, Full Test Set, Threshold = 2.5

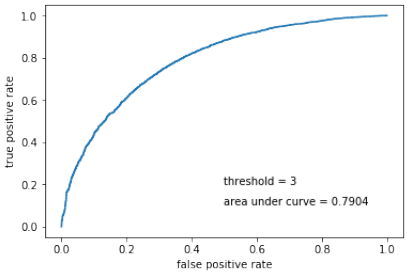

Fig. 29. MAE, MF with Bias, Full Test Set, Threshold = 3

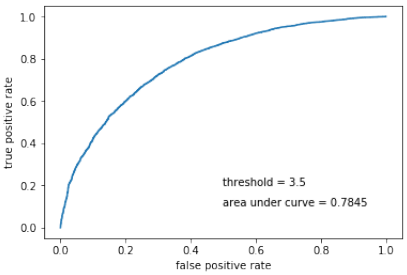

Fig. 30. RMSE, MF with Bias, Full Test Set, Threshold = 3.5

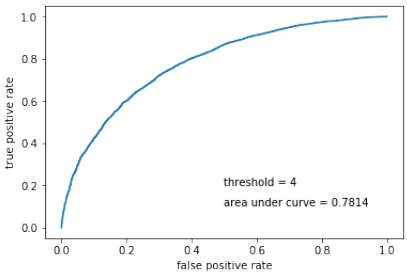

Fig. 31. MAE, MF with Bias, Full Test Set, Threshold = 4

Threshold:	2.5	3	3.5	4
Area Under Curve:	0.7801	0.7904	0.7845	0.7814

Table 2. AUC for Threshold Values, MF with Bias

6 Naive Collaborative Filtering

Although models can be used to represent a distribution of users and ratings, they can be computation heavy and take many hours to train. Does the quality of their predictions outweigh the cost of fitting the model? In order to investigate this question, a light-weight naive filter is compared to the performance of the other models.

6.1 Prediction Function

The naive filter we will quantify is one which takes the mean rating value of a user and provides this as the prediction for a movie the user has not rated. In other words, the following prediction function is used for every i user and j movie, where μ_i is the mean rating for every user i: $\hat{r_{ij}} = \mu_i$

6.2 Design and Test via Cross-validation

The Naive Collaborative filter was designed using the fit-predict paradigm commonly adapted by other machine learning models. Notice that the model is fit to the data once during creation, and never during the cross-validation process.

```
class Naive():
mu_i = None

def __init__(self, X_train):
    self.mu_i = np.zeros(np.mean(X_train, axis=1).shape)
    self.fit(X_train, None, enabled=True)

# calculate mu_i for training
def fit(self, X, y, enabled=False):
    if enabled:
        for idx, row in enumerate(X):
            self.mu_i[idx] = np.mean(row[np.nonzero(row)])

# return mu_i as rating prediction (for each movie)
def predict(self, X, inds=None):
    if inds is None:
        return np.repeat(np.reshape(self.mu_i,
        ↪  (self.mu_i.shape[0], 1)), X.shape[1], axis=1)
    else:
        # return a specific set of mu_i's:
        return np.repeat(np.reshape(self.mu_i[inds],
        ↪  (self.mu_i[inds].shape[0], 1)), X.shape[1], axis=1)
```

Question 30 The average RMSE of the naive filter tested on the entire dataset using cross-fold validation is **0.9373**.

6.3 Naive Collaborative Filter Performance on Trimmed Test Set

Question 31 The average RMSE of the naive filter tested on the popular test set using cross-fold validation is **0.9346**.

Question 32 The average RMSE of the naive filter tested on the unpopular test set using cross-fold validation is **0.9922**.

Question 33 The average RMSE of the naive filter tested on the high-variance test set using cross-fold validation is **1.5123**.

6.4 Naive Collaborative Filter performance analysis

The RMSE of the popular test set is lower than the overall RMSE. This can be explained by the fact that movies which have a lot of ratings will have more accurate means than movies which have fewer ratings. The RMSE of the unpopular test set is higher than the RMSE of the popular test set for this same reason. The RMSE of the high-variance test set is very high because movies which have a high variance will have ratings which are very far away from the mean rating.

Although this collaborative filter is very easy to understand and fast to train, it does not perform well with movies which have few ratings, movies which have many ratings that vary per user, and movies which have multiple centroids that cannot be accurately represented by a scalar mean calculation. In special cases, a naive collaborative filter may be a practical tool to model data and improve prediction speeds. However, more complex models are required for our case.

7 Performance Comparison

Question 34 From fig 32, we conclude that MF with bias performs the best as it has the highest area under the curve and k-NN performs the worst as it has the least area under the curve. Apart from that, all the three algorithms follow similar trends and have very similar results.

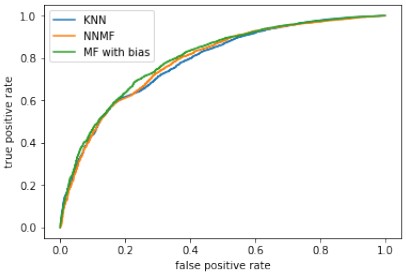

Fig. 32. Precision-Recall curve for k-NN, NNMF and MF with bias

8 Ranking

In this section, we:

- Compute the predicted ratings for all items using one of the collaborative filtering algorithm for each user and store it as a list.
- Then we sort this list in descending order and get the item which has the highest predicted ratings at the top of the list and the item with the lowest predicted ratings at the bottom.
- Based on the value of t, we select the first t-items from the sorted list and then recommend it to the user.

8.1 Evaluating ranking using precision-recall curve

Let G be the set of items liked by the user. These are our ground-truth positives. Let's say we recommend this user items of size t denoted by the set $s(t)$. In this set, we drop the items whose ground truth rating is not known. Using the above two sets,

$$Precision(t) = \frac{|S(t) \cap G|}{|S(t)|} \tag{2}$$

$$Recall(t) = \frac{|S(t) \cap G|}{|G|} \tag{3}$$

Question 35 Apart from using the the two equations described in the last section, we can also use the following formulae:

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

Where TP is the number of True Positives, FP is the number of True Negatives and FN is the number of False Negatives.

Hence, we can define precision as the proportion of positive identifications that were actually correct while Recall is the proportion of actual positives that were also identified correctly.

For our ranking algorithms, Precision is the proportion of items in the set that we have recommended to the user which are actually relevant and Recall is the proportion of relevant items in the set that we have recommended which are are also a part of the first k recommendations.

Question 36 In this section, fig 33 is the plot of average precision (along y-axis) against t (x-axis), fig 34 is the plot of average recall (along y-axis) against t (x-axis) and fig 35 is the plot of average precision (along y-axis) against average recall (x-axis) for k-NN collaborative filter. We select the 'minimum k' that we found in question 11. From the figures, it is clear that as t increases, the precision decreases while the recall increases. These results are in accordance with the equations for precision and recall that we have discussed in the previous sections.

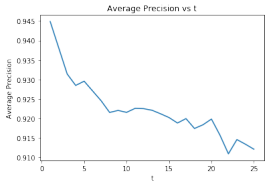

Fig. 33. Average Precision vs t for k-NN

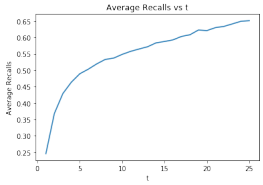

Fig. 34. Average Recall vs t for k-NN

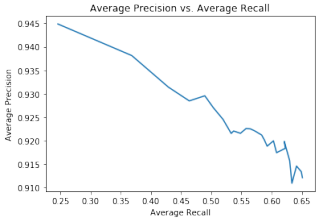

Fig. 35. Average Precision vs Average Recall for k-NN

Question 37 In this section, fig 36 is the plot of average precision (along y-axis) against t (x-axis), fig 37 is the plot of average recall (along y-axis) against t (x-axis) and fig 38 is the plot of average precision (along y-axis) against average recall (x-axis) for NNMF collaborative filter. We select the 'minimum k' that we found in question 18. From the figures, it is clear that as t increases, the precision decreases while the recall increases. These results are in accordance with the equations for precision and recall that we have discussed in the previous sections.

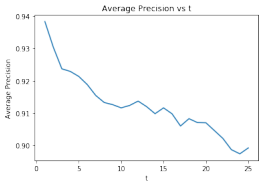

Fig. 36. Average Precision vs t for NNMF

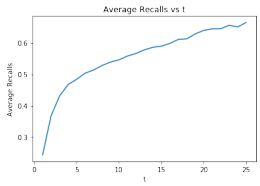

Fig. 37. Average Recall vs t for NNMF

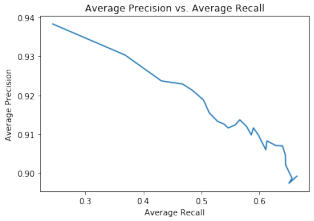

Fig. 38. Average Precision vs Average Recall for NNMF

Question 38 In this section, fig 33 is the plot of average precision (along y-axis) against t (x-axis), fig 34 is the plot of average recall (along y-axis) against t (x-axis) and fig 35 is the plot of average precision (along y-axis) against average recall (x-axis) for MF with bias collaborative filter. We select the 'minimum k' that we found in question 25. From the figures, it is clear that as t increases, the precision decreases while the recall increases. These results are in accordance with the equations for precision and recall that we have discussed in the previous sections.

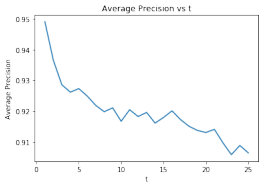

Fig. 39. Average Precision vs t for MF with bias

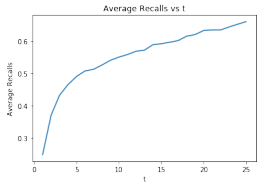

Fig. 40. Average Recall vs t for MF with bias

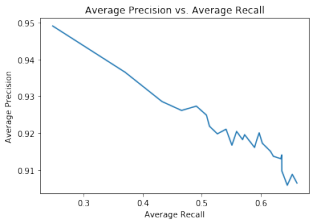

Fig. 41. Average Precision vs Average Recall for MF with bias

Question 39 From the plot in fig 42, we notice that all the three algorithms have very similar trends for the given dataset. However, k-NN and MF with bias perform slightly better than NNMF for some values. We can see that both k-NN and MF with bias keep going up and down and hence we cannot say for certain which one will perform better for all values.

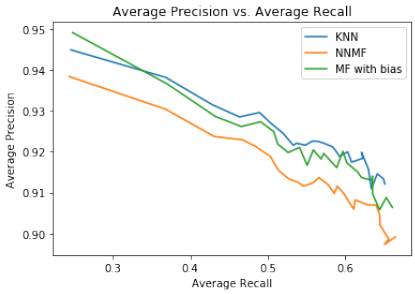

Fig. 42. Precision-Recall curve for k-NN, NNMF and MF with bias

9 Conclusion

By incorporating ratings from multiple users, collaborative filter models can be used as powerful recommender tools. In this project, we showed how different models can be used as collaborative filters. These models included k-NN, NNMF, MF with bias, and a naive-based approach. We found that the naive collaborative filter, performed exceptionally fast however it was bad at generalization. We have shown that the k-NN, NNMF, and MF with bias models are capable of recommending new movies to users.

Bibliography

[Roy19] Roychowdhury, Vwani: Project 3: Collaborative Filtering University of California, Los Angeles, 2019