Project 3: Recommender Systems

[illegible] 8th, 2022 by 11:59 pm

1 Introduction

The increasing impor[illegible]s a medium for electronic and business transactions and advertisement, an[illegible] served as a driving force behind the development of recommender systems technology. Among the benefits, recommender systems provide a means to prioritize data for each user from the infinite information available on the internet. Such systems are critical to ensuring (among others): (a) the detection of hate speech, (b) user retention on a web service, and (c) fast and high-quality access to relevant information. An important catalyst is the ease with which the web enables users to provide feedback about a small portion of the web that they traverse.

Such user-driven *sparse* feedback poses the following challenge in the desing of recommender systems: Can we utilize these sparse user datapoints to infer generalized user interests?

We define some terms:

- The entity to which the recommendation is provided is referred to as the *user*;
- The product being recommended is an *item*.

The basic models for recommender systems works with two kinds of data:

A User-Item interactions such as ratings (a user (you) provides ratings about a movie (item));

B Attribute information about the users and items such as textual profiles or relevant keywords (deep representations about a user or item).

Models that use type A data are referred to as collaborative filtering methods, whereas models that use type B data are referred to as content-based methods. In this project, we will build a recommendation system using collaborative filtering methods.

2 Collaborative filtering models

Collaborative filtering models use the collaborative power of established user-item interactions to make recommendations about new user-item interactions. In this project we use a ratings database where the user is an audience member who viewed a movie, and the item is the movie being rated.

The main challenge in designing collaborative filtering methods is that the underlying ratings matrices are sparse. Consider this example of a movie application in which users specify ratings indicating their like or dislike of specific movies. *Most users would have viewed only a small fraction of the large universe of available movies and as a result most of the ratings are unspecified.*

> ***The basic idea of collaborative filtering methods is that these unspecified ratings can be imputed because the observed ratings are often highly correlated across various users and items.***

For example, consider two users named John and Molly, who have very similar tastes. If their respective ratings exist within our database and are very similar, then the media recommended to them should likely be similar as well.

For those few scenarios in which only John has rated a movie M, the similarity across other movies to Molly's preferences should make clear that Molly might also prefer movie M. Thus, most collaborative filtering methods leverage either inter-item correlations or inter-user correlations for the prediction process.

In this project, we will implement and analyze the performance of two types of collaborative filtering methods:

1. **Neighborhood-[...]ve filtering**: Directly leverages the choices of other users to determin[...] recommend to the current user.
2. **Model-based c[...]ing**: Estimates a joint model from all the user data such that in orde[...] recommendation, we do not need to use the entire user base and ca[...] model.

3 Dataset

In this project, we will build a recommendation system to predict the ratings of movies in the provided dataset. The dataset can be downloaded using the following link: `https://drive.google.com/drive/folders/1_JF9plSjE3PAFBuSyUFRkDdftJWo1TFz?usp=sharing`.

For the subsequent discussion, we assume that the ratings matrix is denoted by R (you will have to construct this), and it is an $m \times n$ matrix containing m users (rows) and n movies (columns). The (i, j) entry of the matrix is the rating by user i for movie j and is denoted by r_{ij}. Before moving on to the collaborative filter implementation, we will analyze and visualize some properties of this dataset.

QUESTION 1: Explore the Dataset: In this question, we explore the structure of the data.

A **Compute the sparsity of the movie rating dataset**:

$$Sparsity = \frac{\text{Total number of available ratings}}{\text{Total number of possible ratings}} \tag{1}$$

B **Plot a histogram showing the frequency of the rating values**: Bin the raw rating values into intervals of width 0.5 and use the binned rating values as the horizontal axis. Count the number of entries in the ratings matrix R that fall within each bin and use this count as the height of the vertical axis for that particular bin. Comment on the shape of the histogram.

C **Plot the distribution of the number of ratings received among movies**: The X-axis should be the movie index ordered by decreasing frequency and the Y-axis should be the number of ratings the movie has received; ties can broken in any way. A monotonically decreasing trend is expected.

D **Plot the distribution of ratings among users**: The X-axis should be the user index ordered by decreasing frequency and the Y-axis should be the number of movies the user has rated. The requirement of the plot is similar to that in Question C.

E **Discuss the salient features of the distributions** from Questions C,D and their implications for the recommendation process.

F **Compute the variance of the rating values received by each movie**: Bin the variance values into intervals of width 0.5 and use the binned variance values as the horizontal axis. Count the number of movies with variance values in the binned intervals and use this count as the vertical axis. Briefly comment on the shape of the resulting histogram.

4 Neighborhood-based collaborative filtering

The basic idea in neighborhood-based methods is to use either user-user similarity or item-item similarity to make predictions from a ratings matrix. There are two basic principles used in neighborhood-based models:

1. *User-based mode*[illegible]ave similar ratings on the same item. Therefore, if John and Molly l[illegible]n a similar way in the past, then one can use John's observed ratings [illegible]*inator* to predict Molly's rating on this movie. *Item is kept constant.*

2. *Item-based model*[illegible] rated in a similar way by the same user. Therefore, John's ratings or[illegible]tion movies like *Alien* and *Predator* can be used to predict his rating on *Terminator*. *User is kept constant.*

In this project, we will only implement user-based collaborative filtering (implementation of item-based collaborative filtering is very similar).

4.1 User-based neighborhood models

In this approach, we are trying to find a set of users similar in their rating strategy to a target user. This results in a user-based neighborhood and we will use the majority vote within the neighborhood to provide recommendations.

In order to determine the neighborhood of the target user u, their similarity to all the other users is computed. Therefore, a similarity function needs to be created between each pair of the historical rating patterns - one by each user across the movies. In this project, we will use the Pearson-correlation coefficient to compute this similarity as a correlation.

4.2 Pearson-correlation coefficient

The Pearson-correlation coefficient between users u and v denoted by Pearson(u,v) captures the similarity between the rating vectors of users u and v. First some notation:

- I_u : Set of item indices for which ratings have been specified by user u;
- I_v : Set of item indices for which ratings have been specified by user v;
- μ_u: Mean rating for user u computed using her specified ratings;
- r_{uk}: Rating of user u for item k.

QUESTION 2: Understanding the Pearson Correlation Coefficient:

A Write down the formula for μ_u in terms of I_u and r_{uk};

B In plain words, explain the meaning of $I_u \cap I_v$. Can $I_u \cap I_v = \emptyset$? (Hint: Rating matrix R is sparse)

Then, with the above notation, the Pearson-correlation coefficient between a pair of users u and v is defined by equation 2:

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

4.3 k-Nearest neighborhood (k-NN)

Having introduced a similarity metric between users (as a correlation coefficient between their ratings across movies), we are now ready to define a neighborhood of users. The k-Nearest neighbors of user u, denoted by P_u, is the set of k users with the highest Pearson-correlation coefficient with user u (pairwise).

4.4 Prediction function

The predicted rating t[illegible]ard for item j, denoted by $\hat{r}_{uj}$, can simply be modeled by equation 3:

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{v \in P_u} Pearson(u, v)(r_{vj} - \mu_v)}{\sum_{v \in P_u} |Pearson(u, v)|} \tag{3}$$

QUESTION 3: Understanding the Prediction function: Can you explain the reason behind mean-centering the raw ratings $(r_{vj} - \mu_v)$ in the prediction function? (Hint: Consider users who either rate all items highly or rate all items poorly and the impact of these users on the prediction function.)

4.5 k-NN collaborative filter

The previous sections have equipped you with the basics needed to implement a k-NN collaborative filter for predicting ratings of the movies. *Although we have provided you with the equations needed to write a function for predicting the ratings, we don't require you to write it. Instead, you can use the built-in python functions for prediction.*

4.5.1 Design and test via cross-validation

In this part of the project, you will design a k-NN collaborative filter and test its performance via 10-fold cross validation. In a 10-fold cross-validation, the dataset is partitioned into 10 equal sized subsets. Of the 10 subsets, a single subset is retained as the validation data for testing the filter, and the remaining 9 subsets are used to train the filter. The cross-validation process is then repeated 10 times, with each of the 10-subsets used exactly once as the validation data.

QUESTION 4: Design a k-NN collaborative filter to predict the ratings of the movies in the original dataset and evaluate its performance using 10-fold cross validation. Sweep k (number of neighbors) from 2 to 100 in step sizes of 2, and for each k compute the average RMSE and average MAE obtained by averaging the RMSE and MAE across all 10 folds. Plot average RMSE (Y-axis) against k (X-axis) and average MAE (Y-axis) against k (X-axis).

The functions that might be useful for solving this question are described in these documentations [1].

Use Pearson-correlation function as the similarity metric. You can read about how to specify the similarity metric in the documentation: `http://surprise.readthedocs.io/en/stable/similarities.html`

QUESTION 5: Use the plot from question 4, to find a 'minimum k'. Note: The term 'minimum k' in this context means that increasing k above the minimum value would not result in a significant decrease in average RMSE or average MAE. If you get the plot correct, then 'minimum k' would correspond to the k value for which average RMSE and average MAE converges to a steady-state value. Please report the steady state values of average RMSE and average MAE.

[1] `http://surprise.readthedocs.io/en/stable/knn_inspired.html`, `http://surprise.readthedocs.io/en/stable/model_selection.html#surprise.model_selection.validation.cross_validate`

4.6 Filter performance on trimmed test set

In this part of the project, we will analyze the performance of the k-NN collaborative filter in predicting the ratings of the movies in the trimmed test set. The test set can be trimmed in many ways, but we will consider the following trimming options:

- Popular movie trimming: In this trimming, we trim the test set to contain movies that have received more than 2 ratings. If a movie in the test set has received less than or equal to 2 ratings in the entire dataset then we delete that movie from the test set and do not predict the rating of that movie using the trained filter.
- Unpopular movie trimming: In this trimming, we trim the test set to contain movies that have received less than or equal to 2 ratings. If a movie in the test set has received more than 2 ratings in the entire dataset then we delete that movie from the test set and do not predict the rating of that movie using the trained filter.
- High variance movie trimming: In this trimming, we trim the test set to contain movies that have variance (of the rating values received) of at least 2 and has received at least 5 ratings in the entire dataset. If a movie has variance less than 2 or has received less than 5 ratings in the entire dataset then we delete that movie from the test set and do not predict the rating of that movie using the trained filter.

Having defined the types of trimming operations above, now we can evaluate the performance of the k-NN filter in predicting the ratings of the movies in the trimmed test set.

4.6.1 Performance evaluation using ROC curve

Receiver operating characteristic (ROC) curve is a commonly used graphical tool for visualizing the performance of a binary classifier. It plots the true positive rate (TPR) against the false positive rate (FPR).

In the context of recommendation systems, it is a measure of the relevance of the items recommended to the user. Since the observed ratings are in a continuous scale (0-5), so we first need to convert the observed ratings to a binary scale. This can be done by thresholding the observed ratings. If the observed rating is greater than the threshold value, then we set it to 1 (implies that the user liked the item). If the observed rating is less than the threshold value, then we set it to 0 (implies that the user disliked the item). After having performed this conversion, we can plot the ROC curve for the recommendation system in a manner analogous to that of a binary classifier.

QUESTION 6: For EACH of the 3 subsets in the test set, design:
A k-NN collaborative filter to predict the ratings of the movies in the test subset (i.e Popular, Unpopular or High-Variance) and evaluate each of the three models' performance using 10-fold cross validation:

- Sweep k (number of neighbors) from 2 to 100 in step sizes of 2, and for each k compute the average RMSE obtained by averaging the RMSE across all 10 folds. Plot average RMSE (Y-axis) against k (X-axis). Also, report the minimum average RMSE.
- Plot the ROC curves for the k-NN collaborative filters for threshold values $[2.5, 3, 3.5, 4]$. For each of the plots, also report the area under the curve (AUC) value.

We provide you with the following hints:

- For each value of k, split the dataset into 10 pairs of training and test sets.

  (trainset 1, testset 1), (trainset 2, testset 2), ..., (trainset 10, testset 10)

  The following documentation might be useful for the splitting:
  `http://surprise.readthedocs.io/en/stable/getting_started.html#use-cross-validation-iterators`
- For each pair of (trainset, testset):

- Train the collaborative filter on the train set
  - Write a trimming function that takes as input the test set and outputs a trimmed test set
  - Predict the ratings of the movies in the trimmed test set using the trained collaborative filter
  - Compute the RMSE of the predictions in the trimmed test set
- Compute the average RMSE by averaging across all the 10 folds.
- For the ROC plotting, split the dataset into 90% for training and 10% for testing. The functions described in the documentation below might be useful http://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html

5 Model-based collaborative filtering

In model-based collaborative filtering, models are developed using machine learning algorithms to predict users' ratings of unrated items. Some examples of model-based methods include decision trees, rule-based models, bayesian methods, and latent factor models. In this project, we will explore latent factor based models for collaborative filtering.

5.1 Latent factor based collaborative filtering

Latent factor based models can be considered as a direct method for matrix completion. It estimates the missing entries of the rating matrix R, to predict what items a user will most probably like other than the ones they have rated. The basic idea is to exploit the fact that significant portions of the rows and columns of the rating matrix are correlated. As a result, the data has built-in redundancies and the rating matrix R can be approximated by a low-rank matrix. The low-rank matrix provides a robust estimation of the missing entries.

The method of approximating a matrix by a low-rank matrix is called matrix factorization. The matrix factorization problem in latent factor based model can be formulated as an optimization problem given by 4

$$\underset{U,V}{\text{minimize}} \quad \sum_{i=1}^{m}\sum_{j=1}^{n}(r_{ij} - (UV^T)_{ij})^2 \tag{4}$$

In the above optimization problem, U and V are matrices of dimension $m \times k$ and $n \times k$ respectively, where k is the number of latent factors. However, in the above setting it is assumed that all the entries of the rating matrix R is known, which is not the case with sparse rating matrices. Fortunately, latent factor model can still find the matrices U and V even when the rating matrix R is sparse. It does it by modifying the cost function to take only known rating values into account. This modification is achieved by defining a weight matrix W in the following manner:

$$W_{ij} = \begin{cases} 1, r_{ij} \text{ is known} \\ 0, r_{ij} \text{ is unknown} \end{cases}$$

Then, we can reformulate the optimization problem as

$$\underset{U,V}{\text{minimize}} \quad \sum_{i=1}^{m}\sum_{j=1}^{n}W_{ij}(r_{ij} - (UV^T)_{ij})^2 \tag{5}$$

Since the rating matrix R is sparse, the observed set of ratings is very small. As a result, it might cause over-fitting. A common approach to address this problem is to use regularization. The optimization problem with regularization is given by equation 6. The regularization parameter λ is always non-negative and it controls the weight of the regularization term.

$$\underset{U,V}{\text{minimize}} \quad \sum_{i=1}^{m}\sum_{j=1}^{n}W_{ij}(r_{ij} - (UV^T)_{ij})^2 + \lambda\,\|U\|_F^2 + \lambda\,\|V\|_F^2 \tag{6}$$

There are many variations to the unconstrained matrix factorization formulation (equation 6) depending on the modification to the objective function and the constraint set. In this project, we will explore two such variations:

- Non-negative matrix factorization (NMF)
- Matrix factorization (MF) with bias)

5.2 Non-negative matrix factorization (NMF)

Non-negative matrix factorization may be used for ratings matrices that are non-negative. As we have seen in the lecture, the major advantage of this method is the high level of interpretability it provides in understanding the user-item interactions. The main difference from other forms of matrix factorization is that the latent factors U and V must be non-negative. Therefore, optimization formulation in non-negative matrix factorization is in 7:

$$\begin{aligned} \underset{U,V}{\text{minimize}} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} W_{ij}(r_{ij} - (UV^T)_{ij})^2 + \lambda \|U\|_F^2 + \lambda \|V\|_F^2 \\ \text{subject to} \quad & U \geq 0, V \geq 0 \end{aligned} \tag{7}$$

There are many optimization algorithms like stochastic gradient descent (SGD), alternating least-squares (ALS),etc for solving the optimization problem in 7. Since you are familiar with the SGD method, we will not describe it here. Instead, we will provide the motivation and main idea behind the ALS algorithm. SGD is very sensitive to initialization and step size. ALS is less sensitive to initialization and step size, and therefore a more stable algorithm than SGD. ALS also has a faster convergence rate than SGD. The main idea in ALS, is to keep U fixed and then solve for V. In the next stage, keep V fixed and solve for U. In this algorithm, at each stage we are solving a least-squares problem.

Although ALS has a faster convergence rate and is more stable, we will use SGD in this project. The main reason behind this is based on the fact that the python package that we will be using to design the NMF-based collaborative filter only has the SGD implementation. This choice would have no effect on the performance of the filter designed because both the SGD and ALS converges for the original dataset. The only downside of using SGD is that it will take a little bit longer to converge, but that will not be a big issue as you will see while designing the NMF filter.

QUESTION 7: **Understanding the NMF cost function:** Is the optimization problem given by equation 5 convex? Consider the optimization problem given by equation 5. For U fixed, formulate it as a least-squares problem.

5.2.1 Prediction function

After we have solved the optimization problem in equation 7 for U and V, then we can use them for predicting the ratings.The predicted rating of user i for item j, denoted by $\hat{r}_{ij}$, is given by equation 8

$$\hat{r}_{ij} = \sum_{s=1}^{k} u_{is} \cdot v_{js} \tag{8}$$

Having covered the basics of matrix factorization, now we are ready to implement a NMF based collaborative filter to predict the ratings of the movies. We have provided you with the necessary background to implement the filter on your own, but we don't require you to do that. Instead, you can use provided functions in Python for the implementation.

5.2.2 Design and test via cross-validation

In this part, you will design a NMF-based collaborative filter and test its performance via 10-fold cross validation. Details on 10-fold cross validation have been provided in one of the earlier sections.

QUESTION 8: Desi[illegible]ollaborative Filter:

A Design a NMF-ba[illegible]ter to predict the ratings of the movies in the original dataset and evalu[illegible] using 10-fold cross-validation. Sweep k (number of latent factors) fro[illegible]es of 2, and for each k compute the average RMSE and average MAE obt[illegible]the RMSE and MAE across all 10 folds. If NMF takes too long, you can [illegible]. Increasing it too much will result in poorer granularity in your results. Plot the average RMSE (Y-axis) against k (X-axis) and the average MAE (Y-axis) against k (X-axis). For solving this question, use the default value for the regularization parameter.

B Use the plot from the previous part to find the optimal number of latent factors. Optimal number of latent factors is the value of k that gives the minimum average RMSE or the minimum average MAE. Please report the minimum average RMSE and MAE. Is the optimal number of latent factors same as the number of movie genres?

C **Performance on trimmed Test set subsets**: For each of Popular, Unpopular and High-Variance test subsets -

- Design a NMF collaborative filter to predict the ratings of the movies in the trimmed test subset and evaluate it's performance using 10-fold cross validation. Sweep k (number of latent factors) from 2 to 50 in step sizes of 2, and for each k compute the average RMSE obtained by averaging the RMSE across all 10 folds.
- Plot average RMSE (Y-axis) against k (X-axis); item Report the minimum average RMSE.

- Plot the ROC curves for the NMF-based collaborative filter designed in part A for threshold values $[2.5, 3, 3.5, 4]$. For the ROC plotting use the optimal number of latent factors found in question B. For each of the plots, also report the area under the curve (AUC) value.

For solving this question, the functions described in the documentation below might be useful: http://surprise.readthedocs.io/en/stable/matrix_factorization.html

5.2.3 Interpretability of NMF

The major advantage of NMF over other forms of matrix factorization is not necessarily one of accuracy, but that of the high level of interpretability it provides in understanding user-item interactions. In this part of the project, we will explore the interpretability of NMF. Specifically, we will explore the connection between latent factors and movie genres.

QUESTION 9: Interpreting the NMF model: Perform Non-negative matrix factorization on the ratings matrix R to obtain the factor matrices U and V, where U represents the user-latent factors interaction and V represents the movie-latent factors interaction (use $k = 20$). For each column of V, sort the movies in descending order and report the genres of the top 10 movies. Do the top 10 movies belong to a particular or a small collection of genre? Is there a connection between the latent factors and the movie genres?

In this question, there will be 20 columns of V and you don't need to report the top 10 movies and genres for all the 20 columns. You will get full credit, as long as you report for a couple columns and provide a clear explanation on the connection between movie genres and latent factors.

5.3 Matrix factorization with bias (MF with bias)

In MF with bias, we modify the cost function (equation 6) by adding bias term for each user and item. With this modification, the optimization formulation of MF with bias is given by equation 9

$$\underset{U,V,b_u,b_i}{\text{minimize}} \quad [illegible])^2 + \lambda \|U\|_F^2 + \lambda \|V\|_F^2 + \lambda \sum_{u=1}^{m} b_u^2 + \lambda \sum_{i=1}^{n} b_i^2 \tag{9}$$

In the above formulat[illegible] of user u and b_i is the bias of item i, and we jointly optimize over $U, V, b_u,$ [illegible]al values.

5.3.1 Prediction func[illegible]

After we have solved the optimization problem in equation 9 for U, V, b_u, b_i, then we can use them for predicting the ratings. The predicted rating of user i for item j, denoted by $\hat{r}_{ij}$ is given by equation 10

$$\hat{r}_{ij} = \mu + b_i + b_j + \sum_{s=1}^{k} u_{is} \cdot v_{js} \tag{10}$$

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.

5.3.2 Design and test via cross-validation

In this part, you will design a MF with bias collaborative filter and test it's performance via 10-fold cross validation. Details on 10-fold cross validation have been provided in one of the earlier sections.

QUESTION 10: Designing the MF Collaborative Filter:

A Design a MF-based collaborative filter to predict the ratings of the movies in the original dataset and evaluate it's performance using 10-fold cross-validation. Sweep k (number of latent factors) from 2 to 50 in step sizes of 2, and for each k compute the average RMSE and average MAE obtained by averaging the RMSE and MAE across all 10 folds. Plot the average RMSE (Y-axis) against k (X-axis) and the average MAE (Y-axis) against k (X-axis). For solving this question, use the default value for the regularization parameter.

B Use the plot from the previous part to find the optimal number of latent factors. Optimal number of latent factors is the value of k that gives the minimum average RMSE or the minimum average MAE. Please report the minimum average RMSE and MAE. Is the optimal number of latent factors same as the number of movie genres?

C **Performance on Test set subsets**: For each of Popular, Unpopular and High-Variance test subsets -

  - Design a MF collaborative filter to predict the ratings of the movies in the trimmed test subset and evaluate it's performance using 10-fold cross validation. Sweep k (number of latent factors) from 2 to 50 in step sizes of 2, and for each k compute the average RMSE obtained by averaging the RMSE across all 10 folds.
  - Plot average RMSE (Y-axis) against k (X-axis); item Report the minimum average RMSE.

- Plot the ROC curves for the NMF-based collaborative filter designed in part A for threshold values $[2.5, 3, 3.5, 4]$. For the ROC plotting use the optimal number of latent factors found in question B. For each of the plots, also report the area under the curve (AUC) value.

For solving this question, the functions described in the documentation below might be useful: `http://surprise.readthedocs.io/en/stable/matrix_factorization.html`

6 Naive collaborative filtering

In this part of the project, we will implement a naive collaborative filter to predict the ratings of the movies in the original dataset. This filter returns the mean rating of the user as it's predicted rating for an item.

6.1 Prediction fur

The predicted rating o denoted by $\hat{r}_{ij}$ is given by equation 11

$$\hat{r}_{ij} = \mu_i \tag{11}$$

where μ_i is the mean

6.2 Design and test via cross-validation

Having defined the prediction function of the naive collaborative filter, we will design a naive collaborative filter and test it's performance via 10-fold cross validation.

An important thing to note about the naive collaborative filter is that there is no notion of training. For training the model, split the dataset into 10 pairs of train set and test set and for each pair predict the ratings of the movies in the test set using the prediction function (no model fitting required). Then compute the RMSE for this fold and repeat the procedure for all the 10 folds. The average RMSE is computed by averaging the RMSE across all the 10 folds.

QUESTION 11: **Designing a Naive Collaborative Filter:**

- Design a naive collaborative filter to predict the ratings of the movies in the original dataset and evaluate it's performance using 10-fold cross validation. Compute the average RMSE by averaging the RMSE across all 10 folds. Report the average RMSE.
  Note that in this case, when performing the cross-validation, there is no need to calculate μ_i's for the training folds each time. You are only asked to use a single set of μ_i's calculated on the entire dataset and validate on 10 validation folds.
- **Performance on Test set subsets:** For each of Popular, Unpopular and High-Variance test subsets -
  - Design a naive collaborative filter to predict the ratings of the movies in the popular movie trimmed test set and evaluate it's performance using 10-fold cross validation.
  - Compute the average RMSE by averaging the RMSE across all 10 folds. Report the average RMSE.

7 Performance comparison

In this section, we will compare the performance of the various collaborative filters (designed in the earlier sections) in predicting the ratings of the movies in the original dataset.

QUESTION 12: **Comparing the most performant models across architecture:** Plot the best ROC curves (threshold = 3) for the k-NN, NMF, and MF with bias based collaborative filters in the same figure. Use the figure to compare the performance of the filters in predicting the ratings of the movies.

8 Ranking

Two primary ways in [illegible] may be formulated:

1. *Prediction*: Predict the rating for a user-item combination;
2. *Ranking*: Recomm[illegible]ns for a particular user.

In previous parts of [illegible] explored collaborative filtering techniques for solving the prediction version [illegible] this part, we will explore techniques for solving the ranking version of the [illegible] two approaches to solve the ranking problem:

- Design algorithm[illegible]nking problem directly
- Solve the predicti[illegible] then rank the predictions

Since we have already solved the prediction problem, so for continuity we will take the second approach to solving the ranking problem.

8.1 Ranking predictions

The main idea of the second approach is that it is possible to rank all the items using the predicted ratings. The ranking can be done in the following manner:

- For each user, compute it's predicted ratings for all the items using one of the collaborative filtering techniques. Store the predicted ratings as a list L.
- Sort the list in descending order, the item with the highest predicted ratings appears first and the item with the lowest predicted ratings appears last.
- Select the first t-items from the sorted list to recommend to the user.

8.2 Evaluating ranking using precision-recall curve

Precision-recall curve can be used to evaluate the relevance of the ranked list. Before stating the expressions for precision and recall in the context of ranking, let's introduce some notation:

$S(t)$: The set of items of size t recommended to the user. In this recommended set, ignore (drop) the items for which we don't have a ground truth rating.

G: The set of items liked by the user (ground-truth positives)

Then with the above notation, the expressions for precision and recall are given by equations 8 and 9 respectively

$$Precision(t) = \frac{|S(t) \cap G|}{|S(t)|} \tag{12}$$

$$Recall(t) = \frac{|S(t) \cap G|}{|G|} \tag{13}$$

QUESTION 13: Understanding Precision and Recall in the context of Recommender Systems: Precision and Recall are defined by the mathematical expressions given by equations 12 and 13 respectively. Please explain the meaning of precision and recall in your own words.

Both precision and recall are functions of the size of the recommended list (t). Therefore, we can generate a precision-recall plot by varying t.

QUESTION 14: Comparing the precision-recall metrics for the different models:

- For each of the three architectures:
  - Plot average precision (Y-axis) against t (X-axis) for the ranking obtained using the model's predictions.
  - Plot the average recall (Y-axis) against t (X-axis) and plot the average precision (Y-axis) against avera[illegible]ll (X-axis).
  - Use the best k [illegible]us parts and sweep t from 1 to 25 in step sizes of 1. For each plot, bri[illegible] shape of the plot.
- Plot the best prec[illegible]btained for the three models (Naive, NMF, MF) in the same figure. Use [illegible]re the relevance of the recommendation list generated using k-NN, NMF[illegible]predictions.

Hints:

- Use threshold = 3 for obtaining the set G
- Use 10-fold cross-validation to obtain the average precision and recall values for each value of t. To be specific, compute precision and recall for each user using equations 12 and 13 and then average across all the users in the test set to obtain the precision and recall for this fold. Now repeat the above procedure to compute the precision and recall for all the folds and then take the average across all the 10-folds to obtain the average precision and average recall value for this value of t.
- If $|G| = 0$ for some user in the test set, then drop this user
- If some user in the test set has rated less than t items, then drop this user

Submission

Please submit your **report**, and your **codes** with a **readme file** on how to run your code to Gradescope/BruinLearn. Only one submission per team is required. If you have any questions you can contact the TAs or post on Piazza.