MODULE 4

Evaluating Recommender Systems

Here, we discuss about various mechanisms for evaluating various recommendation algorithms.

A proper design of the evaluation system is crucial in order to obtain an understanding of the effectiveness of various recommendation algorithms. An incorrect design of the experimental evaluation can lead to either gross underestimation or overestimation of the true accuracy of a particular algorithm or model.

Evaluation Paradigms

There are three primary types of evaluation of recommender systems:

1. User studies
2. Online evaluations
3. Offline evaluations with historical data sets.

The first two types involve users, although they are conducted in slightly different ways. The main differences between the first two settings lie in how the users are recruited for the studies. Although online evaluations provide useful insights about the true effects of a recommendation algorithm, there are often significant practical impediments in their deployment.

1. User Studies
   - In user studies, test subjects are actively recruited, and they are asked to interact with the recommender system to perform specific tasks.
   - Feedback can be collected from the user before and after the interaction, and the system also collects information about their interaction with the recommender system.
   - These data are then used to make inferences about the likes or dislikes of the user. For example, users could be asked to interact with the recommendations at a product site and give their feedback about the quality of the recommendations.
   - An important advantage of user studies is that they allow for the collection of information about the user interaction with the system.
   - Various scenarios can be tested about the effect of changing the recommender system on the user interaction, such as the effect of changing a particular algorithm or user-interface.
   - On the other hand, the active awareness of the user about the testing of the recommender system can often bias her choices and actions.
   - It is also difficult and expensive to recruit large cohorts of users for evaluation purposes. In many cases, the recruited users are not representative of the general population because the recruitment process is itself a bias-centric filter, which cannot be fully controlled.

2. Online Evaluation
   - Online evaluations also leverage user studies except that the users are often real users in a fully deployed or commercial system.
   - This approach is sometimes less susceptible to bias from the recruitment process, because the users are often directly using the system in the natural course of affairs.
   - Typically, users can be sampled randomly, and the various algorithms can be tested with each sample of users.
   - A typical example of a metric, which is used to measure the effectiveness of the recommender system on the users, is the conversion rate.

- The conversion rate measures the frequency with which a user selects a recommended item. For example, in a news recommender system, one might compute the fraction of times that a user selects a recommended article. If desired, expected costs or profits can be added to the items to make the measurement sensitive to the importance of the item. These methods are also referred to as A/B testing, and they measure the direct impact of the recommender system on the end user.
- The basic idea in these methods is to compare two algorithms as follows:
  - ✓ Segment the users into two groups A and B.
  - ✓ Use one algorithm for group A and another algorithm for group B for a period of time, while keeping all other conditions (e.g., selection process of users) across the two groups as similar as possible.
  - ✓ At the end of the process, compare the conversion rate (or other payoff metric) of the two groups.
- The main disadvantage is that such systems cannot be realistically deployed unless a large number of users are already enrolled. Therefore, it is hard to use this method during the start up phase.

3. Offline Evaluation with Historical Data Sets
   - In offline testing, historical data, such as ratings, are used.
   - In some cases, temporal information may also be associated with the ratings, such as the time-stamp at which each user has rated the item.
   - A well known example of a historical data set is the Netflix Prize data set . This data set was originally released in the context of an online contest, and has since been used as a standardized benchmark for testing many algorithms.
   - The main advantage of the use of historical data sets is that they do not require access to a large user base.
   - Once a data set has been collected, it can be used as a standardized benchmark to compare various algorithms across a variety of settings.
   - Furthermore, multiple data sets from various domains (e.g., music, movies, news) can be used to test the generalizability of the recommender system.
   - Offline methods are among the most popular techniques for testing recommendation algorithms, because standardized frameworks and evaluation measures have been developed for such cases.
   - The main disadvantage of offline evaluations is that they do not measure the actual propensity of the user to react to the recommender system in the future. For example, the data might evolve over time, and the current predictions may not reflect the most appropriate predictions for the future.

General Goals of Evaluation Design

1. *Accuracy*
   - Accuracy is one of the most fundamental measures through which recommender systems are evaluated.

2. *Coverage*
   - Even when a recommender system is highly accurate, it may often not be able to ever recommend a certain proportion of the items, or it may not be able to ever recommend to a certain proportion of the users.
   - This measure is referred to as coverage.

- For example, in a rating matrix contains a single entry for each row and each column, then no meaningful recommendations can be made by almost any algorithm.
- There are two types of coverage, which are referred to as user-space coverage and item-space coverage.
- User-space coverage measures the fraction of users for which at least k ratings may be predicted.
- The value of k should be set to the expected size of the recommendation list. When fewer than k ratings can be predicted for a user, it is no longer possible to present a meaningful recommendation list of size k to the user.
- Itemspace coverage measures the fraction of items for which the ratings of at least k users can be predicted.
- A different form of item-space coverage evaluation is defined by the notion of catalog coverage, which is specifically suited to recommendation lists.

3. *Confidence and Trust*
    - The estimation of ratings is an inexact process that can vary significantly with the specific training data at hand.
    - Furthermore, the algorithmic methodology might also have a significant impact on the predicted ratings.
    - This always leads to uncertainty in the user about the accuracy of the predictions. Many recommender systems may report ratings together with confidence estimates.
    - For example, a confidence interval on the range of predicted ratings may be provided. In general, recommender systems that can accurately recommend smaller confidence intervals are more desirable because they bolster the user's trust in the system.
    - The algorithm with the smaller confidence interval width will win as long as both algorithms are correct (i.e., within the specified intervals) at least 95% of the time on the hidden ratings. If one of the algorithms falls below the required 95% accuracy, then it automatically loses. Unfortunately, if one system uses 95% confidence intervals and another uses 99% confidence intervals, it is not possible to meaningfully compare them. Therefore, it is possible to use such systems only by setting the same level of confidence in both cases.
    - ***While confidence measures the system's faith in the recommendation, trust measures the user's faith in the evaluation.***
    - Trust measures the level of faith that the user has in the reported ratings.
    - Even if the predicted ratings are accurate, they are often not useful if the user fails to trust the provided ratings.

4. *Novelty*
    - The novelty of a recommender system evaluates the likelihood of a recommender system to give recommendations to the user that they are not aware of, or that they have not seen before.
    - Unseen recommendations often increase the ability of the user to discover important insights into their likes and dislikes that they did not know previously.

5. *Serendipity*
    - The word "serendipity" literally means "lucky discovery."

- Therefore, serendipity is a measure of the level of surprise in successful recommendations.
- In other words, recommendations need to be unexpected.
- In contrast, novelty only requires that the user was not aware of the recommendation earlier.
- Serendipity is a stronger condition than novelty.
- All serendipitious recommendations are novel, but the converse is not always true.
- Consider the case where a particular user frequently eats at Indian restaurants. The recommendation of a new Pakistani restaurant to that user might be novel if that user has not eaten at that restaurant earlier. However, such a recommendation is not serendipitious, because it is well known that Indian and Pakistani food are almost identical. On the other hand, if the recommender system suggests a new Ethiopian restaurant to the user, then such a recommendation is serendipitious because it is less obvious.
- There are two methods for evaluating serendipity:
  1. Online methods: The recommender system collects user feedback both on the usefulness of a recommendation and its obviousness. The fraction of recommendations that are both useful and non-obvious, is used as a measure of the serendipity.
  2. Offline methods: One can also use a primitive recommender to generate the information about the obviousness of a recommendation in an automated way. The primitive recommender is typically selected as a content-based recommender, which has a high propensity for recommending obvious items. Then, the fraction of the recommended items in the top-k lists that are correct (i.e., high values of hidden ratings), and are also not recommended by the primitive recommender are determined. This fraction provides a measure of the serendipity.

6. *Diversity*
   - The notion of diversity implies that the set of proposed recommendations within a single recommended list should be as diverse as possible.
   - For example, consider the case where three movies are recommended to a user in the list of top-3 items.
   - If all three movies are of a particular genre and contain similar actors, then there is little diversity in the recommendations. If the user dislikes the top choice, then there is a good chance that she might dislike all of them.
   - Presenting different types of movies can often increase the chance that the user might select one of them.
   - Ensuring greater diversity can often increase the novelty and serendipity of the recommendations. Furthermore, greater diversity of recommendations can also increase the sales diversity and catalog coverage of the system.
   - Diversity can be measured in terms of the content-centric similarity between pairs of items.
   - For example, if a set of k items are recommended to the user, then the pairwise similarity is computed between every pair of items in the list.
   - The average similarity between all pairs can be reported as the diversity.
   - Lower values of the average similarity indicate greater diversity. Diversity can often provide very different results from those of accuracy metrics.

7. *Robustness and Stability*

- A recommender system is stable and robust when the recommendations are not significantly affected in the presence of attacks such as fake ratings or when the patterns in the data evolve significantly over time.
- For example, the author or publisher of a book might enter fake positive ratings about a book at Amazon.com, or they might enter fake negative ratings about the books of a rival.

8. *Scalability*
    - In recent years, it has become increasingly easy to collect large numbers of ratings and implicit feedback information from various users.
    - In such cases, the sizes of the data sets continue to increase over time.
    - As a result, it has become increasingly essential to design recommender systems that can perform effectively and efficiently in the presence of large amounts of data.
    - A variety of measures are used for determining the scalability of a system:
        a. Training time:
            - ✓ Most recommender systems require a training phase, which is separate from the testing phase.
            - ✓ In most cases, the training is done offline. Therefore, as long as the training time is of the order of a few hours, it is quite acceptable in most real settings.
        b. Prediction time:
            - ✓ Once a model has been trained, it is used to determine the top recommendations for a particular customer.
            - ✓ It is crucial for the prediction time to be low, because it determines the latency with which the user receives the responses.
        c. Memory requirements:
            - ✓ When the ratings matrices are large, it is sometimes a challenge to hold the entire matrix in the main memory.
            - ✓ In such cases, it is essential to design the algorithm to minimize memory requirements.
            - ✓ When the memory requirements become very high, it is difficult to use the systems in large-scale and practical settings.

Design Issues in Offline Recommender Evaluation

- It is crucial to design recommender systems in such a way that the accuracy is not grossly.
- A common mistake made by analysts in the benchmarking of recommender systems is to use the same data for parameter tuning and for testing.
- Such an approach grossly overestimates the accuracy because parameter tuning is a part of training, and the use of test data in the training process leads to overfitting.
- To guard against this possibility, the data are often divided into three parts:

    a)Training data:

- ✓ This part of the data is used to build the training model. For example, in a latent factor model, this part of the data is used to create the latent factors from the ratings matrix. One might even use these data to create multiple models in order to eventually select the model that works best for the data set at hand.

b) Validation data:

- ✓ This part of the data is used for model selection and parameter tuning.
- ✓ For example, the regularization parameters in a latent factor model may be determined by testing the accuracy over the validation data. In the event that multiple models have been built from the training data, the validation data are used to determine the accuracy of each model and select the best one.

c)Testing data:

- ✓ This part of the data is used to test the accuracy of the final (tuned) model.
- ✓ It is important that the testing data are not even looked at during the process of parameter tuning and model selection to prevent overfitting. The testing data are used only once at the very end of the process.

Case Study of the Netflix Prize Data Set

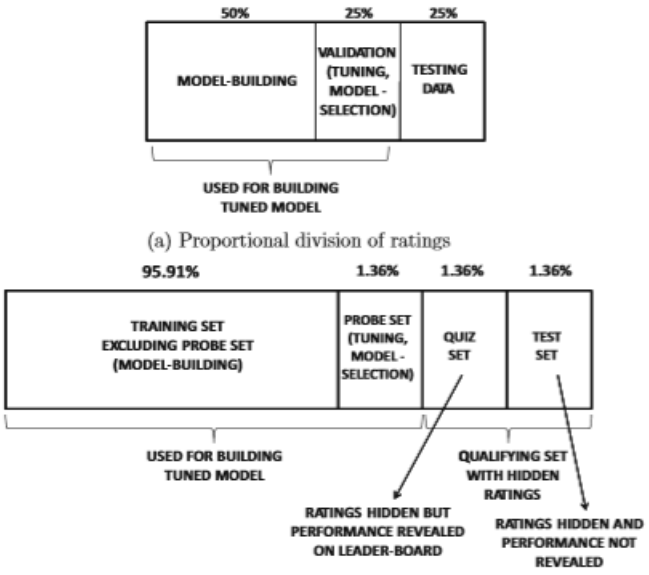

Figure 7.1: Partitioning a ratings matrix for evaluation design

- In the Netflix data set, the largest portion of the data set contained 95.91% of the ratings.
- This portion of the data set was typically used by the contest participants for model-building.
- Another 1.36% of the data set was revealed to the participants as a probe set. Therefore, the model-building portion of the data and the probe data together contained 95.91 + 1.36 = 97.27% of the data.
- The probe set was typically used by contests for various forms of parameter tuning and model selection, and therefore it served a very similar purpose as a validation set.

- The probe and qualifying sets were based on more recent ratings, compared to the 95.91% of the ratings in the first part of the training data.
- The ratings of the remaining 2.7% of the data were hidden, and only triplets of the form (User,Movie,GradeDate) were supplied without actual ratings.
- The main difference from a test set was that participants could submit their performance on the qualifying set to Netflix.
- The performance on half the qualifying data, known as the quiz set, was revealed to the participants on a leader-board.
- The overall division of the Netflix data set is shown in Figure 7.1(b).
- The only difference from the division in Figure 7.1(a) is the presence of an additional quiz set.

Segmenting the Ratings for Training and Testing

The entries of the rating matrix are divided into training and testing data using two methods.

- Hold-out
- Cross validation

The hierarchical division is illustrated in Figure 7.2. In the following, we will consistently use the terminology of the first level of division in Figure 7.2 into "training" and "testing" data, even though the same approach can also be used for the second level division into model building and validation portions. This consistency in terminology is followed to avoid confusion.

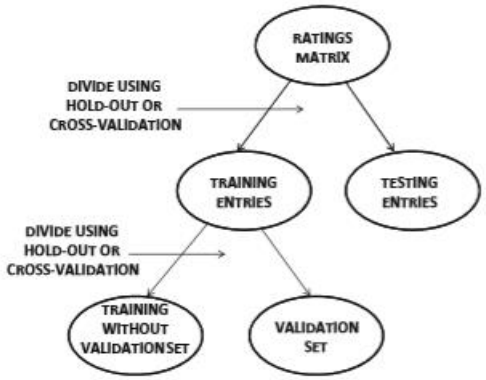

Figure 7.2: Hierarchical division of rated entries into training, validation, and testing portions

a)Hold-Out

- ✓ In the hold-out method, a fraction of the entries in the ratings matrix are hidden, and the remaining entries are used to build the training model.
- ✓ The accuracy of predicting the hidden entries is then reported as the overall accuracy. S
- ✓ Such an approach ensures that the reported accuracy is not a result of overfitting to the specific data set, because the entries used for evaluation are hidden during training. Such an approach, however, underestimates the true accuracy.

b)Cross-Validation

- ✓ In the cross-validation method, the ratings entries are divided into q equal sets.
- ✓ Therefore, if S is the set of specified entries in the ratings matrix R, then the size of each set, in terms of the number of entries, is |S|/q.
- ✓ One of the q segments is used for testing, and the remaining (q −1) segments are used for training.
- ✓ In other words, a total of |S|/q entries are hidden during each such training process, and the accuracy is then evaluated over these entries.
- ✓ This process is repeated q times by using each of the q segments as the test set.
- ✓ A special case is one where q is chosen to be equal to the number of specified entries in the ratings matrix.
- ✓ Therefore, |S|−1 rating entries are used for training, and the one entry is used for testing.
- ✓ This approach is referred to as leave-one-out cross-validation.

Accuracy Metrics in Offline Evaluation

- Offline evaluation can be performed by measuring the accuracy of predicting rating values (e.g., with RMSE) or by measuring the accuracy of ranking the recommended items.
- In the Netflix Prize competition, the RMSE measure was used for final evaluation.

Measuring the Accuracy of Ratings Prediction

Once the evaluation design for an offline experiment has been finalized, the accuracy needs to be measured over the test set. As discussed earlier, let S be the set of specified (observed) entries, and $E \subset S$ be the set of entries in the test set used for evaluation. Each entry in E is a user-item index pair of the form (u, j) corresponding to a position in the ratings matrix. Note that the set E may correspond to the held out entries in the hold-out method, or it may correspond to one of the partitions of size $|S|/q$ during cross-validation.

Let r_{uj} be the value of the (hidden) rating of entry $(u, j) \in E$, which is used in the test set. Furthermore, let $\hat{r}_{uj}$ be the predicted rating of the entry (u, j) by the specific training algorithm being used. The entry-specific error is given by $e_{uj} = \hat{r}_{uj} - r_{uj}$. This error can be leveraged in various ways to compute the overall error over the set E of entries on which the evaluation is performed. An example is the *mean squared error*, denoted by *MSE*:

$$MSE = \frac{\sum_{(u,j) \in E} e_{uj}^2}{|E|} \tag{7.4}$$

Clearly, smaller values of the *MSE* are indicative of superior performance. The square-root of this value is referred to as the *root mean squared error (RMSE)*, and it is often used instead of the *MSE*.

$$RMSE = \sqrt{\frac{\sum_{(u,j) \in E} e_{uj}^2}{|E|}} \tag{7.5}$$

The *RMSE* is in units of ratings, rather than in units of squared ratings like the *MSE*. The *RMSE* was used as the standard metric for the Netflix Prize contest. One characteristic of the *RMSE* is that it tends to disproportionately penalize large errors because of the squared term within the summation. One measure, known as the *mean-absolute-error (MAE)*, does not disproportionately penalize larger errors:

$$MAE = \frac{\sum_{(u,j) \in E} |e_{uj}|}{|E|} \tag{7.6}$$

Other related measures such as the normalized *RMSE* (*NRMSE*) and normalized *MAE* (*NMAE*) are defined in a similar way, except that each of them is divided by the range $r_{max} - r_{min}$ of the ratings:

$$NRMSE = \frac{RMSE}{r_{max} - r_{min}}$$

$$NMAE = \frac{MAE}{r_{max} - r_{min}}$$

The normalized values of the *RMSE* and *MAE* always lie in the range $(0, 1)$, and therefore they are more interpretable from an intuitive point of view. It is also possible to use these values to compare the performance of a particular algorithm over different data sets with varying scales of ratings.

RMSE versus MAE

- As the RMSE sums up the squared errors, it is more significantly affected by large error values or outliers.
- A few badly predicted ratings can significantly ruin the RMSE measure. In applications where robustness of prediction across various ratings is very important, the RMSE may be a more appropriate measure.
- On the other hand, the MAE is a better reflection of the accuracy when the importance of outliers in the evaluation is limited.
- The main problem with RMSE is that it is not a true reflection of the average error, and it can sometimes lead to misleading results.
- Clearly, the specific choice should depend on the application at hand.