Chalmers Tekniska Högskola

CHALMERS
UNIVERSITY OF TECHNOLOGY

DESIGN OF AI SYSTEMS (DAT410)

Module 2: Recommendation systems

Group 38

Himanshu Sahni
9812075498
sahni@chalmers.se
MPCAS

Nina Uljanić
9609134920
uljanic@chalmers.se
MPCAS

We hereby declare that we have both actively participated in solving every exercise. All solutions are entirely our own work, without having taken part in other solutions.

Hours:
Himanshu Sahni: 18h
Nina Uljanić: 17h

1. Reading and reflection

Problem formulation:

Read the papers "The Netflix" and "Lessons from the Netflix Prize Challenge" and write a half-page summary of your takeaways from them. Pick two design features (e.g., regarding data, models, algorithms) that characterize the task and the solution in the papers and discuss how they may differ in another application of recommendation systems.

Reflection - Himanshu Sahni

- **The Netflix Prize**
  The paper "The Netflix Prize" tells us about the challenge posed by Netflix for the machine learning, data mining, and computer science communities with the aim of developing a recommender system that can beat the accuracy of its Cinematic recommendation system. The Cinematic recommendation system of Netflix used a variant of Pearson's correlation to analyze the accumulated movie ratings weekly to determine similar movies that could be of interest to the user. It also computed a multivariate regression to determine a unique personalized prediction. The performance of the system was measured using RMSE (Root mean Square Error). The objective of the contest was to build a system that can improve the value of RMSE by 10 %. The paper highlights the different techniques used by the teams and the percentage improvement in performance. It was found that the most suitable technique in terms of performance was SVD (Singular Value Decomposition).

- **Lessons from the Netflix Prize Challenge**
  The paper "Lessons from the Netflix prize Challenge" explains the methodology used by the winning team to solve the posed problem. They used two techniques while building their model i.e., KNN (k-nearest neighbors) and latent factor models with SVD (singular value decomposition). Some of the advancements implemented were also discussed, improving the neural network for the calculation of interpolation weights and the regularisation technique.
  The AT&T labs used a hybrid strategy to achieve the improvement of 8.43%. However, the improvement was only 5.10% when using a pure latent factor model and even less in the case of the pure nearest neighbor method.
  KNN captures the pair of items that are related to each other with some similarities in order to predict the ratings of an unrated item based on these similarities by a user. It ignores the values that are not in its closed neighborhood, which may be of interest to the user. On the other hand, the latent factor model with SVD captures the most prominent features and ignores the relations among small sets of closely related movies. Both of these methods address quite different levels of structure in the data and that's why they complement each other.
  Three approaches to improve the existing methods were also highlighted:

  1. Deepening Known Methods
  2. Combining multiscale views of data
  3. Combining Explicit rating information with implicit rating behavior

Discussion for two Design Features

The two design features that are used in building this model are latent factor models and K nearest neighbor. With these methods, there was an improvement in the RMSE score. However, it is important to note that the accuracy of the model was majorly dependent on user input or feedback. In recommender systems, the objective is to predict user behavior and for this, a wide variety of techniques can be applied but there is a lot of dependency on the data, its form, type, quality, and structure can affect the effectiveness of the method. For example, it is mentioned that implicit techniques should be used to gather data such as purchase history, rental history, and browsing patterns. This can help in gaining and enhancing user preferences without actually engaging them. However, it is believed that a combined input of the explicit and implicit behavior is the most desirable. Adding complementary data viewpoints to the model such as local and binary views can also help in increasing the accuracy.

Reflection - Nina Uljanić

Discussion

The Netflix Prize competition had one goal: to produce a 10% reduction in the root mean squared error of test data when compared to Netflix's own algorithm, Cinematch. In other words, they were looking to improve the accuracy of the recommender system by 10%. An immense dataset - with more than 100 million ratings from 480 000 users on nearly 18 thousand movies - was published by Netflix for this challenge, and it has inadvertently led to advancements in the field of collaborative filtering, the main technique used in recommender systems.

The goal of 10% improvement was not reached; the results were not too far off, however, with the greatest attained reduction of the root mean square error, or RMSE, used to determine the accuracy of the system, was set at 8.43%. In order to achieve this, an ensemble of complementary models was selected. Among these models were nearest neighbor models (k-NN) and latent factor models (SVD, RBM), alongside custom algorithms/models by the team. The benefit of using multiple models is that they can complement the shortcomings of each other, creating a more robust, efficient system. By using various models as standalone solutions, they could reach 5-7% of error reduction, compared of 8.43% when using a combination of models. As the authors of the paper *Lessons from the Netflix Prize Challenge* say, "(models) address quite different levels of structure in the data."

The k-NN model is most effective at detecting very localized relationships. It identifies pairs of items that are similar, i.e. tend to be rated similarly, and then uses this information to predict ratings of a yet unrated item based on similar items (by the same user). It is notable that the model did not use all of the data provided, but only a small portion of it - about 20 to 50 similar (neighboring) items - for any single prediction. This method has its downsides as well, namely that it is unable to take the complete information contained in the dataset. This means that the recommendation system might not have a full understanding of the user's preferences.

Latent factor models, on the other hand, work to uncover hidden characteristics that explain the nuances in the ratings, that is, it is used to understand the underlying relationships between different items and users. This model complements the k-NN model and its lack of understanding of the relationship between items and users. Such models fail to classify movies that are a part of the same series, which is in fact a strong suit of the neighbor-based models described above.

In conclusion, a system made up of a variety of complementing models which utilize both implicit and explicit input excels at the job of finding personalized recommendations.

The two models mentioned above, k-NN and latent models, are well established and are widely used for making movie recommendations. However, the performance of these methods in other recommendation systems can vary, and may depend on several factors, such as the type of data being used, the complexity of the relationships between items and users, and the specific requirements of the recommendation system.

For example, in a music recommendation system, where the underlying relationships between songs and users may be less easy to understand, k-NN may be a better choice because it is able to make recommendations based on the nearest neighbors of a given song or user. Unlike with movies, where users' preferences can be described in terms of many quantifiable characteristics, such as movie's genre, director, actors, etc., music is a more subjective and personal experience, and the factors that influence a user's preferences for different songs may be more complex and difficult to identify, rendering the latent factor models inapplicable. A user's mood, location, and cultural background may all play a role in determining their preferences for different songs.

One can see that k-NN models are used for recommending music in popular audio streaming services - if one listens to any given genre for a period of time, the system will recommend only similar music, and rarely, if ever, will it recommend something "different."

Latent factor models, on the other hand, would be useful in recommending news articles, as news articles often have well-defined attributes such as the topics they cover, the publication date, the author, etc. The factors that influence a user's preference for different types of news, for example their political beliefs, can be used to recommend a variety of relevant readings from various sources (this is important, especially when covering society-related issues). A k-NN model would not be as useful in this scenario, as it might recommend articles on limited topics and from a selected collection of sites, potentially dangerously narrowing the user's worldview.

2. Implementation

Problem formulation:

In your report, describe the system, and motivate the choices you made. Comment on the strength and weaknesses of the system. Suggest 5 movies for each of the first 5 users, Vincent, Edgar, Addilyn, Marlee, and Javier, that they haven't rated already.

Solution:

We explored multiple approaches while solving this problem. We tried a few different implementations, however, they didn't seem to work well with this dataset. One of the major problems that we faced while working with this dataset is the limitation of the information. The other datasets explored by us had a lot of other parameters like cast, crew, description of the movie, etc. Also, for this dataset, we have a lot of missing data, which makes it a little difficult compared to the other datasets that we reviewed.

We have implemented the truncated SVD matrix decomposition method from the sci-kit learn library. Truncated Singular Value Decomposition (SVD) is a matrix factorization technique that factors a matrix M into the three matrices U, Σ, and V. This is very similar to PCA, excepting that the factorization for SVD is done on the data matrix, whereas for PCA, the factorization is done on the covariance matrix. Typically, SVD is used under the hood to find the principle components of a matrix.(**Source: scikit-learn**). It is important to note that it reduces the size of the data i.e., the dimensionality is reduced and this reduced matrix can then be used as a proxy.
In the given dataset we have 2000 movies, 600 users, and 25 different movie genres. First of all, we have filtered the data and removed some of the columns. We need to input the user movie reviews matrix as our required data along with the parameters of the truncated SVD algorithm i.e., epsilon and latent factors. This introduces a bias and the number of features we want to extract from the user movie reviews matrix. For our case, the latent factor is 60. Finally, we get the user preference matrix and movie-feature matrix.

The advantage of the truncated SVD algorithm is that it is easy to implement since it has very few parameters.

One of the disadvantage is that the resulting matrix has a lot of negative values, therefore we need to use a suitable normalization method.

Using the prediction rating matrix generated with Truncated SVD, here are the 5 movie recommendations for the following users:

User	Movies
Vincent	1. The Good Thief 2. Maximum Risk 3. Seeking a Friend for the End of the World 4. Much Ado About Nothing 5. The longest Ride
Edgar	1. Lars and the Real Girl 2. Theresa Is a Mother 3. 500 Days of Summer 4. Drop Dead Gorgeous 5. Wild Things
Addylin	1. Perrier's Bounty 2. Igby Goes Down 3. Dysfunctional Friends 4. 9 5. ATL
Marlee	1. Perrier's Bounty 2. Zero Effect 3. Adventureland 4. Dear John 5. Interstellar
Javier	1. Now You See Me 2 2. Homefront 3. Flyboys 4. Sonny with a Chance 5. Bran Nue Dae

3. Discussion

Problem formulation:

Assessing the quality of a recommendation system before deploying it to users is difficult. Why? In a few paragraphs, discuss fundamental challenges in evaluation of recommendation systems and how they may lead to problems in practice.

Discussion

The key to creating a successful recommendation system is having high-quality data to serve as a foundation for the system. The data is received from the users in the form of feedback (ex. ratings). A system that is about to be deployed might not have access to high-quality data; instead, the data used might be incomplete. This lack of data can lead to the system generating inaccurate recommendations.

The system might not be efficient until it gets to know the user; the more feedback a user gives, the more the system will improve. Preferences are subjective; it is impossible to know whether a user would like the recommended item before he/she evaluates it.

A system might be hard to evaluate because of limited data available about users and their preferences. It is hard to know whether the system works as intended, i.e. it will improve with more data, or whether the implementation is simply subpar.

Another challenge is user engagement; if users do not give feedback on the items they have tried after being recommended the item, then the data will stay limited and the efficiency of the system cannot be determined.

Although the overall amount of data accessible in recommender systems is frequently large, this does not always imply that the data is relevant or that it can be processed or converted into meaningful data. This can be roughly described as "the ratio of the relevant data to the total amount of data available in the selected proxies." From this perspective, the following problems could develop:

- If there is an imbalance between these two sets, some algorithms may be unable to accurately identify certain attributes, which would reduce the performance of the recommendation system.
- It is also possible to have a situation where the user's preferences are not closely related, for example, when the "ratio of available meaningful data in the selected proxies and the total amount of data" is low. As a result, a collaborative filtering approach could also produce incorrect recommendations with a high-quality score for any given selected metric.
- Depending on the aspects related to the proxies, the scope of a proxy or proxies defined to build the recommender system may have various consequences during the deployment phase, different behaviors may be inferred or extracted, and it is also possible that some behaviors may be ignored. It might be possible to design better recommendation systems if we use simple proxies to increase the scope of behaviors in comparison to a more complex proxy.

Individual Summary and Reflection

Lecture 2: Recommender Systems (2023-01-24)

Lecture summary by Himanshu Sahni

Recommender Systems are widely used all over the internet for suggesting products or services to users. For example, Netflix uses them to learn user behavior and then suggest movies, e-commerce websites use them to suggest products based on their searches, etc. in general these are the algorithms that are used to suggest a user relevant material(movies to watch, products to buy, text to read). Recommender systems basically work by storing the user preferences and then predicting and suggesting them as the future preference for the user. A Recommender system built can have different objectives i.e., maximizing profits, maximizing the engagement time of the users, engaging more users, etc. To fulfill this objective they will be collecting data either actively or passively and usually the aim is to maximize the objective. Proxies are used in recommendation systems which basically means what can we measure and optimize, for example, user ratings can be used as proxies. A proxy chosen should be possible to measure, possible to be optimized, and should be related to the user behavior. Usually, machine learning techniques are used for this and the two most widely used approaches are content-based filtering and collaborative filtering algorithm.
In content filtering, the idea is to build a model using the available features that explain the user-product interactions. We aim to minimize the error in observed data and this is known as empirical risk minimization. However, one of the major problems while building recommender systems is missing data (users might not rate products) but we can use mean normalization to predict the missing data. To solve the problem of overfitting(low bias and high variance) we use the regularization technique to find the parameters with lower variance.
Collaborative-based methods are based purely on past interactions between users and products. The idea is to use these past interactions between users and products and then predict similar users or products based on these proximities. In collaborative filtering, it is not possible to recommend new users or recommend a new product to the user since we do not have the user-product interaction data for this case, and this problem is known as the cold start problem. In collaborative filtering different matrix decomposition techniques are used to predict the unknown data.
To conclude we can say the wider the scope of the data, the better results (predictions) we can achieve. Also, the recommendations made by the system for one user might not good for the other user, therefore, recommender systems are called personalized machines.

Module 1 reflection by Himanshu Sahni

The previous module gave me a lot of insights into how AI systems can be built. I could understand how important it is to identify the features of a problem. I learned a lot about the different methods for problem-solving or for modeling problems. It is possible to solve a problem with different methods based on the features that we choose. I think the most important learning for me is to emphasize on selecting the right features as it gives a lot of clarity in selecting the ways to model it and this eventually can lead to getting the best results. However, another important learning would be that sometimes the most appropriate solution might not be the optimal one.

Lecture summary by Nina Uljanić

It is well known that companies passively gather data when a user accesses an online service. This collected data can be used to, for example, serve advertisements or recommend products. In this lecture, a bookstore was taken as an example in which recommending products is crucial for business. By analyzing the data of past purchases and ratings, one can tailor book recommendations based on the users' preferences. The chance of an item selling is higher if the user is interested in the item, and maximizing profit is on top of the list of requirements. The lecture considered a scenario in which users with similar interests are compared in order to figure out which item to recommend; if persons A and B have similar tastes, and if person B liked the book X, then it might be worth recommending the book X to user A. Artificial intelligence models can be trained to learn patterns like these. Learning preferences from data can be done through content filtering, which is the most common AI/ML strategy, or collaborative filtering. The former considers the user-product interaction data as well as product features, while the latter works off of user-product interaction data only. A drawback of content filtering is that the data used might be incomplete (for example, most users will not rate every book they buy).

Module 1 reflection by Nina Uljanić

My experience with AI and ML algorithms is limited, but this changed with the first module. The module served as a nice introduction to the field, providing us with a list of the most commonly used methods. Besides understanding the intricacies of the methods, I also had to consider under which circumstances each of the methods may be best suited to solve a problem. Not all methods are equal; they have each their own advantages and disadvantages, and one must understand these if one is to create a usable, high-quality AI system.