Introduction

There are entertainment websites and online stores with millions of items. It becomes challenging for the customer to select the right one. Recommender systems come into the picture and help users to select the right item by suggesting a presumable list of items and find a few specific items which are supposed to be most appealing. It has become an integral part of e-commerce, movie and music rendering sites and the list goes on.

Collaborative Filtering is the most common technique used when it comes to building intelligent recommender systems that can learn to give better recommendations as more information about users is collected. Collaborative Filtering doesn't need anything else except user's historical preference on a set of items. Because it's based on historical data, the core assumption here is that the users who have agreed in the past tend to also agree in the future. There are two ways, in which collaborative filtering runs recommender systems. Memory-based collaborative filtering and model-based collaborative filtering.

The aim of the project was to create different recommender systems which determine a potential rating for a user and movie on the movieLens dataset. MovieLens itself is a research site run by GroupLens Research group at the University of Minnesota. In my project I used small sample of this dataset limited with 600 items. The dataset was split into random train (80%) and test (20%) subsets. Two models (**Item-Based collaborative filtering and SVD**) trained separately on the training dataset and every time a model is learnt, it is used to predict the class of the test dataset.

Item-based collaborative filtering

Memory-based collaborative filtering is a method which uses a rating system to predict the preferences of one user by taking into consideration the preferences of a similar user, or the 'neighbour'. There are two ways to calculate preferences, user-based Collaborative Filtering and item-based Collaborative Filtering. **Item-based collaborative filtering** is an algorithm which identify similar items based on users' previous ratings.

Similarity measures

First step in the item-based collaborative filtering algorithm is to compute the similarity between items and then to select the most similar items. The basic idea in similarity computation between two items *i* and *j* is to first isolate the users who have rated both of these items and then to apply a similarity computation technique to determine the similarity $s_{i,j}$. There are a number of different mathematical formulations that can be used to calculate the similarity between two items. Such as:

- Cosine-Based Similarity
- Correlation-Based Similarity
- Adjusted Cosine Similarity
- 1-Jaccard distance

In my project I used the **Pearson correlation coefficient**. (Correlation-Based Similarity)

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}},$$

This similarity measure is based on how much the ratings by common users for a pair of items deviate from average ratings for those items. The Pearson correlation coefficient, *r*, can take a range of values from +1 to -1. A value of 0 indicates that there is no association between the two variables. A value greater than 0 indicates a positive association and a value less than 0 indicates a negative association. I used the ready function from the sklearn library.

From model to predictions

The goal of a collaborative filtering algorithm is to predict the utility of a certain item for a particular user based on the user's previous ratings of other like-minded users. After building a model by using one of the similarity measures, we can predict the rating for any user-item pair by using the idea of **weighted sum**.

$$P_{u,i} = \frac{\sum_{\text{all similar items, N}}(s_{i,N} * R_{u,N})}{\sum_{\text{all similar items, N}}(|s_{i,N}|)}$$

First we take all the items similar to our target item, and from those similar items, we pick items which the active user has rated. We weight the user's rating for each of these items by the similarity between that and the target item. Finally, we scale the prediction by the sum of similarities to get a reasonable value for the predicted rating.

According to M.A. Ghazanfar and A. Prugel-Bennett [2010] and others conventional weighted sum prediction formula used in item-based collaborative filtering is not correct for very sparse datasets. The similarity between two items can be misleading if they have been rated by very few users. For example, two items can have identical similarity if they have been identically rated by only two users, which is not true. This resulted in bad predictions overall for large test sets.

Factorization method – SVD

The second category covers the model based approaches, which involve a step to reduce or compress the large but sparse user-item matrix. One of the popular algorithms to factorize a matrix is the **singular value decomposition** (SVD) algorithm. SVD came into the light when matrix factorization was seen performing well in the Netflix prize competition. Latent factor models comprise an alternative approach to Collaborative Filtering with the more holistic goal to uncover latent features that explain observed ratings by transforming both items and users to the same latent factor space, thus making them directly comparable. The latent space tries to explain ratings by characterizing both products and users on factors automatically inferred from user feedback.

Factorization

The key idea of SVD models is to factorize the user-item rating matrix to a product of two lower rank matrices. This technique reduces the number of features of a dataset by reducing the space dimension from N-dimension to K-dimension. It uses a matrix structure where each

row represents a user, and each column represents an item. The elements of this matrix are the ratings that are given to items by users.

Singular value decomposition finds factors of matrices from the factorisation of a user-item-rating matrix. The singular value decomposition is a method of decomposing a matrix into three other matrices as given below:

$$A = USV^T$$

Where *A* is a *m x n* utility matrix, *U* is a *m x r* orthogonal left singular matrix, which represents the relationship between users and latent factors, *S* is a *r x r* diagonal matrix, which describes the strength of each latent factor and *V* is a *r x n* diagonal right singular matrix, which indicates the similarity between items and latent factors. The latent factors here are the characteristics of the items, for example, the genre of the music. The SVD decreases the dimension of the utility matrix *A* by extracting its latent factors. It maps each user and each item into a *r*-dimensional latent space. This mapping facilitates a clear representation of relationships between users and items.

$q\ T$' and each user can be represented by a vector 'pu' such that the dot product of those 2 vectors is the expected rating.

$$expected\ rating = \hat{r}_{ui} = q_i^T p_u$$

'q_i^T ' and 'p_u ' can be found in such a way that the square error difference between their dot product and the known rating in the user-item matrix is minimum.

$$minimum(p,q) \sum_{(u,i)\epsilon K} \left(r_{ui} - q_i^T . p_u\right)^2$$

Regularization

For model to be able to generalize well and not over-fit the training set, it is needed to introduce a penalty term to minimization equation. This is represented by a regularization factor λ multiplied by the square sum of the magnitudes of user and item vectors.

$$minimum(p,q) \sum_{(u,i)\epsilon K} \left(r_{ui} - q_i^T . p_u\right)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

Improving SVD - Bias terms

In order to reduce the error between the value predicted by the model and the actual value, the algorithm uses a bias term. Let for a user-item pair *(u, i),* μ is the average rating of all items, 'b_i' is the average rating of item *i* minus μ and 'b_u' is the average rating given by user *u* minus μ, the final equation after adding the regularisation term and bias can be given as:

$$minimum(p, q, b_i, b_u) \sum_{(u,i)\epsilon K} \left(r_{ui} - q_i^T . p_u - \mu - b_i - b_u\right)^2 + \lambda\left(\|q_i\|^2 + \|p_u\|^2 + b_i^2 + b_u^2\right)$$

The above equation is minimized using stochastic gradient descent algorithm.

Challenges

Neighbourhood models are most effective at detecting very localized relationships. They rely on a few significant neighbourhood-relations, often ignoring the vast majority of ratings by a user. Consequently, these methods are unable to capture the totality of weak signals encompassed in all of a user's ratings. Latent factor models are generally effective at estimating overall structure that relates simultaneously to most or all items. However, these models are poor at detecting strong associations among a small set of closely related items, precisely where neighbourhood models do best.

Each method has their own pros and coins however collaborative filtering struggle with some general problems like:

- Sparsity problem. By operating on large quantities of products (counted in millions), the user-item matrix used in the collaborative filtering approach is very rare and the quality of recommendations is questionable.
- There is also a problem with the so-called 'cold start'. It occurs when a new user appears or a new item is added to the catalog. New items cannot be recommended until users have seen movies or rated them. New users cannot receive good quality recommendations because they have made too few ratings or their purchase history is empty.

- Lack of transparency and explainability of this level of information. the underlying tastes expressed by latent features are actually not interpretable because there is no content-related properties of metadata. Users might rate a movie based on their preference for e.g. the actor or soundtrack but collaborative filtering will not take this into account.
- Scalability is another issue, as the Collaborative Filtering algorithm is usually written considering n users. As the data set grows, the overall program can become massive.

Results

There are many evaluation metrics for evaluating recommendation systems. The most popular and most commonly used is RMSE. Precision and recall are classical evaluation metrics in binary classification algorithms and for document retrieval tasks. These metrics have been "Translated" by *turicreate* library to help evaluate recommendation systems.

Results for both applied algorithms are nearly the same.

	Algorythm	RMSE	Recall	Precision
0	Item-Based	1.251672	0.2	0.263805
1	SVD	1.251776	0.2	0.267712

Precision metrics shows that only 26% recommended items that are relevant among all recommended items in both cases. Recall refers to the percentage of of recommended items that are relevant among all relevant items it is only 20% which means both models have a probability of 20% to place an appealing movie to the user. RMSE are also almost identical. Undoubtedly, achieved results are not very good, I assume it is due to the fact that dataset was very sparse.

Conclusion

The above implementation of recommender systems is very basic. Large portals use rather hybrid solutions, trying to estimate our preferences as accurately as possible. Their recommendation methods are based on such large data and are so time-consuming that recommendations are not calculated on an ongoing basis, after each user assessment or after each purchase, and are updated regularly.

This project contains two of the most popular approaches to Collaborative Filtering. First, item-based model, which creates item-item matrix and predicting ratings on items, based on

the similar ratings of users. Second, Singular Value Decomposition (SVD), a method from linear algebra that has been generally used as a dimensionality reduction technique in machine learning. I tried them on the Movielens dataset, but as the results shows, they did not perform very well in testing.

Collaborative Filtering provides strong predictive power for recommender systems. However, it has a several limitations and struggle with some challenges like lack of transparency, 'cold start', sparsity or scalability problem.

Sources

- [2001] B. Sarwar, G. Karypis, J. Konstan, J. Riedl - *Item-based Collaborative Filtering Recommendation Algorithms* Proc. 10th International Conference on the World Wide Web, pp. 285-295.
- [2010] M. Ghazanfar, A. Prugel-Bennett - *Novel Significance Weighting Schemes for Collaborative Filtering: Generating Improved Recommendations in Sparse Environments* , Conference: Proceedings of The 2010 International Conference on Data Mining, DMIN 2010, Las Vegas, Nevada, USA.
- [2010] P. Cremonesi, Y. Koren, R. Turrin - *Performance of Recommender Algorithms on Top-N Recommendation Tasks*, RecSys '10: Proceedings of the fourth ACM conference on Recommender systems, Pages 39–46.
- [2008] Y. Koren - *Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model*, AT&T Labs – Research.