Movie Recommendation in cold start scenario

Saif Masood (1225248414), Vishnusai Kandati(1225463759)
Gayathri Phani Kumar Kanuri(1225608280), Devadutt Sanka(1225362138)

December 7, 2022

Abstract

Recommender Systems are efficient tools for filtering out relevant information. They are employed by various companies to recommend products, movies, news, etcetera to their users. We studied the different recommender systems and the algorithms behind them. Then, using a collaborative filtering method, we created a recommender system that suggests movies to a user. We aim to use a hybrid recommendation model to address the cold start issue that plagues the collaborative filtering technique.

1 Introduction and motivation

Today's web applications deal with an enormous volume of data. Choosing an item from such a massive and varied collection can cause decision paralysis for the user. Recommendation engines can assist users in selecting items that are ideal for them based on their interests and past decisions. Nowadays, large-scale applications like Amazon Prime Video, Netflix, LinkedIn, Instagram, etcetera utilize recommendation engines to improve user engagement and retention. However, it can be challenging to recommend movies to users that are new to the application and have limited interaction data. Thus, our primary goal is to design a recommendation engine that proposes potential movies to users based on their interests, even if the interaction data is sparse. Recommendations by such a system must be better than random.

Recommender Systems that provide potent recommendations when little information about new users and items is available are challenging to build. Collaborative Filtering and Content-based recommendations are the two most popular techniques for building recommendation engines. While collaborative systems account for the aggregate preferences of users having similar interests, content-based filtering primarily employs individual user preferences to suggest items. However, both these techniques perform poorly in the case of cold-start scenarios. Estimating item and user latent factors is challenging when collaborative data is sparse in the case of matrix factorization methods. Content based systems leverage the item metadata to address this issue. As such, recommendations for new items can be generated without having prior collaborative data on these items. However, such CB models fail to leverage any interaction data present and require a large amount of user data to work, thereby performing worse than Collaborative filtering methods in the presence of collaborative information[1].

As we will shortly see, the crux of the recommendation problem lies in maximizing a likelihood function, a recurring theme in our coursework. Asynchronous gradient descent[2] (synchronous technique covered in the course) is employed to train the resulting model using a sigmoid function.

We implement a hybrid recommendation model using the open-source library LightFM [3]. The model performs as well as Content-Based models in cold-start scenarios, outperforming them in the presence of the user and collaborative data. Further, the model performs at least as well as the collaborative models if interaction data is abundant.

2 Problem Description

The cold start problem, in general, concerns the issue that a system cannot draw any useful inferences for items or users if it doesn't have sufficient information about them. It is a well researched problem when it comes to recommender systems. A recommender system typically assesses the user's profile against a set of benchmark qualities. These attributes could be influenced by the item features (content-based filtering) or the user's social context and previous actions (collaborative filtering). The user may be connected to a variety of interactions, depending on the system: ratings, page views, likes, purchases, bookmarks, etc.There are three cases[4] of cold start:

- *New community*
  The new community issue is a system bootstrapping problem that presents itself at the start-up of systems when there is no information available for the recommender to rely on. For instance, although a catalog of items might exist, almost no users are present, and the lack of user interaction makes it very hard to provide reliable recommendations

- *New item*
  The issue that arises when new products added to the catalog that have only a few or no interactions is known as the "item cold-start problem." Since collaborative filtering algorithms rely on an item's interactions to generate recommendations, it is particularly problematic for them. A pure collaborative algorithm cannot recommend items that have no interactions. Even if a collaborative algorithm will recommend it if there are few interactions, the quality of such recommendations will be inadequate. This causes the problem of unpopular items to arise and is unrelated to new products. In some situations (recommendations for movies), a small number of objects may acquire a disproportionately high number of interactions, while the majority of the other items may only receive a small number. This problem is referred to as popularity bias.

- *New user*
  The "new user case" refers to the problem that arises when a new user joins the system and, for a while, the recommender must make recommendations without depending on the user's prior interactions because none have yet taken place. This issue is especially crucial when the recommender is a feature of the service provided to users since poor recommendations may cause the user to discontinue using the system before engaging with it long enough for the recommender to fully grasp his interests. In order to create an initial user profile, it is a common practice to ask new users for their preferences.

Using the hybrid model, we will address the three types of cold-start scenarios.

3 Methodology

The hybrid model that we implemented addresses the following issues:

- The model learns user and item representations using interaction data. For example, if users regularly like objects represented as mobile phones and tabs, the model learns that mobile phones and tabs are similar.

- The model generates recommendations for new users and items.

The latent representation method is used to address the first problem. The embeddings of users who enjoy both mobile phones and tabs will be close; if people who never like both mobile phones and Playstations are the same users, then their embeddings will be extremely different. Tabs can now be suggested to a user who has only used a mobile device.

The second issue is tackled by representing items and users as linear combinations of their content features. Recommendations can be made immediately for new items as the content feature vector is known prior. For instance, the movie "The Shawshank Redemption" is described by features: "Crime", "Morgan Freeman", and "The Shawshank Redemption". The latent representation of

the movie is simply the sum of the features' latent representations.

3.1 The model

The set of users is denoted by U and the set of items is denoted by I. Also, let set F^U represent user features and the set F^I represent item features. We assume that users interact with items positively or negatively. The (user, item) interaction pair is the union of positive (S^+) and negative (S^-) interactions. Each user is represented by a set of features f_u subset of F^U while each item is represented by a set of features f_i subset of F^I. For each feature f, the model is parameterized using d-dimensional user and item feature embeddings, $e^U{}_f$ and $e^I{}_f$. A bias input is also used to describe each feature: $b^f{}_U$ for user and $b^f{}_I$ for item.

The latent representation of a user is given by the sum of it's features' latent representations: The

$$q_u = \sum_{j \in f_u} e_j^u$$

same goes for items:

$$p_i = \sum_{j \in f_i} e_j^i$$

The bias term for a user is given by the sum of the bias terms of all the users features:

$$b_u = \sum_{j \in f_u} b_j^u$$

The same holds for items:

$$b_i = \sum_{j \in f_i} b_j^i$$

The model's prediction for user u and item i is then given by the dot product of user and item representations, adjusted by the user and item feature biases:

$$r_{ui} = f(q_u . p_i + b_u + b_i)$$

$$f(.) = \frac{1}{1 + exp(-x)}$$

Now our optimisation objective simply reduces to maximising the likelihood of data conditional on the parameters:

$$L(e^U, e^I, b^U, b^I) = \prod_{(u,i) \in S^+} r_{ui} \prod_{(u,i) \in S^-} (1 - r_{ui})$$

3.2 Training and dataset

The dataset utilized is the standard MovieLens dataset and the popular Tag Genome tag set[5]. The movie lens dataset includes roughly 10 million movie ratings for 10, 681 films that were provided by 71, 567 people. The Tag Genome tagset is used to get information about the tags and genres of all movies. A relevance score between 0 and 1 is used to represent the aptness of a tag in describing a movie. All ratings above 4.0 (out of 5) are classified as positive and the rest are considered as negative. To keep just the most relevant tags, any tags with a relevance score of less than 0.8 are eliminated. In total, there are 69,878 users, 10 681 items, 9,996,948 interactions, and 1030 unique tags in the final dataset.

The model is trained using asynchronous stochastic gradient descent. We use 4 training threads for the experiment[3]. The per-parameter learning rate schedule is given by Adagrad[6]. The two experiments conducted on the dataset are:

- Warm setting, where an 80-20 training-test split has applied to all items and users represented in the training set.
- The second experiment relates to the Cold-start, in which all interactions for 20 percent

of the items are taken out of the training set and allocated to the test set.

4 Results

The metric Mean Receiver Operating Characteristics Area Under the Curve (ROC AUC), which measures the likelihood that a randomly selected positive item would rank higher than a randomly selected negative item, is used to measure the model's accuracy. A ROC curve is a graph that illustrates how effectively a classification model performs across all thresholds. Two parameters—True Positive Rate and False Positive Rate are plotted on this curve[7]. A high AUC score is similar to a low rank-inversion probability when the recommender inadvertently ranks an ugly item higher than an appealing item[3]. The final score is calculated using the average of this metric across all test participants[3].

Based on the metrics(Figure 1)and graphs(Figure 2), the following results are derived:

1. The model performs at least as well as the CF model when collaborative data is abundant (warm-start, dense user-item matrix).

2. In both cold-start and latent scenarios, the model performs at least as well as pure content-based models, substantially outperforming them when collaborative information is available in the training set.

3. The model can simulate increasingly complex structures and performs better when the latent dimensionality hyperparameter(d) in the cold-start scenario rises.

4. The model performs better than a pure CF in a cold-start scenario.

5. The model performs better than a pure CB model because it leverages collaboration data.

Model	Warm	Cold
Hybrid (tags)	0.74	0.70
Hybrid (tags + id)	0.76	0.71
MF	0.76	0.5

Figure 1: Results of different models with respect to Warm setting and cold start scenario

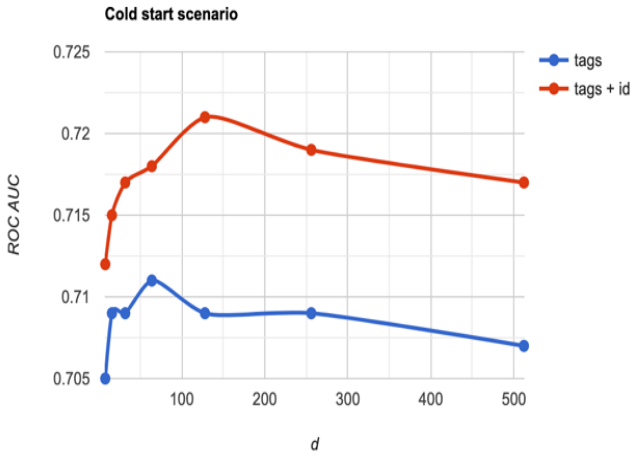

Figure 2: ROC AUC curve values with respect to different latent dimensions(d)

5 Related Work

There have been numerous attempts to use hybrid models to address the cold-start issue by concurrently modifying content and collaborative data. Users are depicted by Soboroff et al. [8] as lin-

ear combinations of feature vectors of objects they have interacted with. To obtain latent user profiles, they next apply LSI on the resulting item-feature matrix. Projecting new things onto the latent feature space allows for the creation of representations of those items. The model's benefit over pure CB techniques is that it makes use of the collaborative information included in the user-feature matrix. Instead of over items, it models user preferences as being determined over certain features (sets of features). Contrary to our hybrid model, which always considers all other features when determining how well a feature predicts an interaction, defining a specific user-item pair.

By using the same item latent feature matrix in both decompositions, Saveski et al. [9] conduct joint factorization of the user-item and item-feature matrices. The parameters are then improved by minimizing a weighted sum of the reproduction loss functions for both matrices. The relative weight of accuracy in the decomposition of the collaborative and content matrices is controlled by a weight hyperparameter. McAuley et al. [10] simultaneously model ratings and product reviews using a similar strategy. Since the user-item matrix factorization is the only optimization goal for our hybrid model in this case, it has the advantage of simplicity.

Shmueli et al. [11] utilize a single-objective strategy, minimize the user-item matrix reproduction loss, and describe items as linear combinations of the latent factors describing their attributes to recommend news articles. In a modified cold-start environment, where both metadata and information about other users who have commented on a particular article are accessible, they demonstrate the viability of their approach. They do not, however, simulate user features, and their methodology does not offer proof of model performance in the warm-start situation. By jointly factorizing the user-item, item-feature, and user-feature matrices, our model adheres to the hybrid model canon. It can be thought of as a specific instance of Factorization Machines from a theoretical perspective[12].

6 Conclusion

In this project, we explored the various techniques for creating a recommendation system. Our main goal was to build a movie recommendation system that addresses the cold-start issue that plagues most traditional Collaborative and Content-based systems. We successfully implemented a hybrid recommendation model that leverages the best of both worlds (content and collaboration) and effectively tackles the cold-start scenario. The next step would be to gauge how well the model generalizes to other data sets like songs, products, news, etcetera. Unlike this attempt, where training the model took hours, we'd partition the data into smaller subsets to quickly assess the performance of the model and any need for parameter tuning. We'd also cast a wider net and explore multiple approaches before drilling down on one.

7 References

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. Knowledge and Data Engineering, IEEE Transactions on, 17(6):734–749, 2005.

[2] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In Advances in Neural Information Processing Systems, pages 693–701, 2011.

[3] Maciej Kula: Metadata Embeddings for User and Item Cold-start Recommendations

[4] Cold start (recommender systems). (2022, July 25). In Wikipedia.

[5] J. Vig, S. Sen, and J. Riedl. The tag genome: Encoding community knowledge to support novel interaction. ACM Transactions on Interactive Intelligent Systems (TiiS), 2, 2012.

[6] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research, 12:2121–2159, 2011

[7] https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

[8] I. Soboroff and C. Nicholas. Combining content and collaboration in text filtering. In Proceedings of the IJCAI, volume 99, pages 86–91, 1999.

[9] M. Saveski and A. Mantrach. Item cold-start recommendations: learning local collective embeddings. In Proceedings of the 8th ACM Conference on Recommender systems, pages 89–96. ACM, 2014.

[10] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In Proceedings of the 7th ACM conference on Recommender systems, pages 165–172. ACM, 2013.

[11] E. Shmueli, A. Kagian, Y. Koren, and R. Lempel. Care to Comment?: Recommendations for commenting on news stories. In Proceedings of the 21st international conference on World Wide Web, pages 429–438. ACM, 2012.

[12] S. Rendle. Factorization machines. In Data Mining (ICDM), 2010 IEEE 10th International Conference on, pages 995–1000. IEEE, 2010