Technology Review: Collaborative filtering algorithms

Junting Wang

Introduction:

In recent years, recommender systems are becoming essential components of the web. Collaborative filtering techniques are among the most successful and applied approaches in the recommendation system. In this technology review, I will be focusing on introducing three of the recent state-of-the-art collaborative filtering models with neural architectures, i.e. NCF, VAE-CF and NGCF, and comparing them in terms of model complexity, performance, and application.

Problem Setup:

The general goal of neural collaborative filtering methods is to learn and model user preferences of items based on implicit feedback. Suppose we use $u \in 1, ..., N$ to index users and $i \in 1, ..., M$ to index items. A user u's preference of an item i is modeled by a score, denoted as Y_{ui}. In general, if Y_{ui} score is high, it is more likely that user u has or will have interactions with item i. All three of the introduced model utilize the user-by-item interaction matrix X to model the implicit feedback, where $X_{ui} = 1$ if an interaction between user u and item i is observed.

Algorithms:

Neural Collaborative Filtering (NCF):

NCF adopts a *multi-layer perceptron* to learn the use-item interaction Y_{ui}. It maps each users and items in to latent space. During training, each entry in the user-item interaction matrix are passed in as user-item pair, i.e. (u,i). The model first gets the latent embedding of both the user and item. The user embedding and item embedding are then fed into a multi-layer neural architecture to map the latent vectors of user u and item i to Y_{ui}, which is viewed as the final preference score for this user-item pair. The predictive model can be formulated as

$$Y_{ui} = f(P^T v_u^U, Q^T v_i^I | P, Q, \theta_f)$$

where $P \in \mathbb{R}^{M*K}$ and $Q \in \mathbb{R}^{N*K}$, denoting the latent embedding for users and items respectively. v_u^U and v_i^I represents the feature vectors of user and item respectively. θ_f represents the model parameters.

NCF explicitly uses user item pairs to model user's preferences. For a given user u, the model essentially goes through all the possible user item pairs of user u and compute the prediction score to model the user's preference of all items, which can be costly.

Neural Graph Collaborative Filtering(NGCF):

In additional to the user-item interaction matrix, NGCF also exploits the user-item graph structure to expressively model the high-order connectivity exists in the graph. The user-item graph is a bipartite graph defined based on user-item interactions. Apart from the latent embedding of items and users, NGCF adopts a *message aggregator* to propagate user-item interactions. The latent vector of a user will consist of both its original latent mapping and the latent aggregation of the connected items

$$e_u = LeakyReLU(e'_u + \sum_{i \in N_u} e'_i)$$

where e'_u and e'_i denotes the original latent user and item embedding respectively. N_u represents the first-order neighbors of user u. Similar to, NCF, NGCF also uses a multi-layer neural architecture to map the latent embedding of user and item into Y_{ui}.

Variational Autoencoders for Collaborative Filtering(VAE-CF):
Unlike the previously introduced models, VAE-CF takes a very different approach in computing and modeling user-item interactions. VAE-CF follows the variational inference training paradigm. The model consists of two parts, encoder and decoder. Instead of using the user-item pair, for user u, during encoding phase, VAE-CF takes the user-item interaction matrix of user u and samples a latent representation of u via a multi-layer perceptron. Then, in the decoding phase, VAE-CF transforms the representation via a non-linear multi-layer neural architecture to produce a probability distribution over the entire item set. Each entry in the decoded distribution is essentially the Y_{ui} that we are trying to compute.

VAE-CF adopts a very different angle to model the user preferences, compared to both NCF and NGCF. It models the user-item interaction matrix as a distribution. The information of user-item pair is implicitly integrated into the constructed distribution, whereas for NCF and NGCF the user-item pair are modeled explicitly.

Comparison:
NCF and NGCF are closely related. NGCF can essentially be viewed as an extension of NCF by capturing high-order connectivity information between users and items. They both models the user-item interaction pair directly. On the other hand, VAE-CF learns the latent representation of the user-item interaction matrix, each user-item pair is implied in such encoding. They all have different training objectives. But the general goals are the same, to get high quality user-item preference score Y_{ui}.

Model Complexity:
Among these three, when fixing the latent vector size and trained on the same data, VAE-CF has the least number parameters to learn, while NGCF has the most parameters. In terms of run time, during training, both NCF and NGCF takes every entry in the user-item matrix as input, whereas VAE-CF encodes the rows of the matrix. Therefore, VAE-CF is the fastest to train, taking significantly less time and resource compared to NCF and NGCF. NGCF, due to the user-item graph aggregation, is the slowest to train. Also, because of the item-user graph, NGCF has scalability issue. It does not scale well for large number of users and items. Sometimes it will not be able to run on large datasets that contains a significant number of user and item.

Performance:
In general, NGCF has the best performance and NCF is the worst among the three. The improvement of NGCF over the NCF can be attributed to the encoded high-order connectivity information. These 3 methods perform better for items with more interactions, and is worse for long-tail items (item with few interactions with users).

Application:
Due to model complexity and scalability, while NGCF gives the best performance, it has limited usage as it will not scale for large datasets. VAE-CF, on the other hand, gives both robust performance, is extremely fast to train and scales well for large datasets. NCF is both expensive to train and gives the

worst performance among the introduced algorithms. In general, VAE-CF is very robust and is more applicable.

Conclusion:

In this technology review, I introduced three state-of-the-art collaborative filtering algorithms with neural architecture. I analyzed each of the three models and talked about the overall architecture and idea. Then, I compared those three models with three different perspectives, i.e. model complexity, performance and application.