Item-based Collaborative Filtering

Summary of paper: Deep Item-based Collaborative Filtering for Top-N Recommendation

Parth Patel | 15th March 2022

Reference - F. Xue, X. He, X. Wang, J. Xu, K. Liu and R. Hong, "Deep Item-based Collaborative Filtering for Top-N Recommendation," presented at the ACM Transactions on Information Systems, Aug 2018

Introduction

- Collaborative Filtering (CF) techniques leverages user-item interaction data to predict user preference. CF is used to select potential candidates for recommending and complemented by ranking engine to rank those candidates.

- There are two types of CF techniques

  - User-based CF (UCF) : In this method the goal is to find similarities among the users using the historical interaction data.

  - Item-based CF (ICF) : In this method the user is represented with the historically interacted items and use the similarity between items to recommend new items.

Why ICF better than UCF?

- ICF represents user with the historically consumed items in contrast to UCF which represents user with an ID. This allows ICF to encode more information in the inputs to the model thus improving both the accuracy and the interpretability of the user preference modeling.

- Second advantage is the composability of ICF in modeling user preferences makes it easy to implement online personalization. For example when user interacts with new item, then instead of re-training model parameters to refresh recommendation list, ICF can approximate the refreshed list by simply retrieving items similar to the new item. In contrast UCF needs to re-train the model parameters to refresh the recommendation list.

ICF Methods Advancement

- Earlier ICF methods used Pearson's correlation and cosine similarity to quantify the similarity between two items. However such methods require extensive manual tuning on the similarity to make them perform well. This makes it difficult to adapt well-tuned method to a new dataset or dataset of new items domain.

- In recent years new data-driven methods are introduced which learn item similarity using the data. The two representative methods are the *sparse linear method* (SLIM) and *factored item similarity model* (FISM).

- However the paper argues that even these methods only model the second-order relations between pairs of items - the relation between an item in user history and the target item. Higher-order relations such as multiple items share certain properties which are likely to be consumed together are not considered.

Paper’s Proposal

- The authors states that HOSLIM (*high order SLIM*) method captures higher-order relations among items into different steps. First it identifies frequent itemsets from user-item interaction and extend SLIM to find *item-itemset similarity matrix* which is then used to find higher-order item relations. Such two step solution requires a support threshold to be tuned to avoid negative effects.

- The paper proposes a method that learns about higher-order item relations in a unified process. The authors leverages the success of neural recommender models and develop a neural network method to achieve the target termed as DeepICF. They conducted experiments on two datasets MovieLens and Pinterest to verify the effectiveness of higher-order relations and further integrate attention design to refine the modeling of second-order item relations which leads to better accuracy and termed it DeepICF+a.

Framework of ICF

- ICF predicts a user-item interaction y_{ui} by assuming that user *u's* preference on item *i* depends on the similarity of *i* to all items that u has interacted with before. In general, the predictive model of ICF can be abstracted as

$$\hat{y}_{ui} = \sum_{j \in R_u^+} s_{ij} r_{uj}$$

where R^+_u is the item set that the user u has interacted with, s_{ij} denotes the similarity between item i and j, and r_{uj} is the u's observed preference on item j, which can be a real-valued rating score (explicit feedback) and a binary 0 or 1 (implicit feedback).

- Clearly, the estimation of item similarity s_{ij} is crucial to the performance of ICF.

Sparse Linear Method (SLIM)

- SLIM is among the first learning-based ICF methods that direct learn the item-item similarity matrix from historical interaction data. Specially, it minimizes the reconstruction error between the original user-item interaction matrix and the reconstructed one that is derived from an ICF model. Two constraints are employed on the item-item similarity matrix: no-negativity and sparsity, which ensure the meaningfulness of the learned similarities and enforce each item to be similar to a few items only. The objective function of SLIM is formulated as below

$$L_{SLIM} = 1/2 \sum_{u=1}^{U} \sum_{i=1}^{I} (r_{ui} - \sum_{j \in R_u^+} s_{ij} r_{uj})^2 + \lambda_1 ||S||_1 + \lambda_2 ||S||_2$$

where U and I denote the number of users and items, $S \in R^{I \times I}$ represents the item-item similarity matrix where each entry is s_{ij} , λ_1 is a hyper-parameter to control the strength of *L1* regularization to enforce the sparsity constraint, and λ_2 is a hyper-parameter to control the strength of *L2* regularization to prevent overfitting.

- In SLIM, item similarity matrix S is the parameter to learn - the constraint $S \geq 0$ is performed to ensure each element in S to be non- negative, and the constraint *diag(S)* = 0 forces the diagonal elements of S to be zero to eliminate the impact the target item itself in estimating a prediction.

Higher-Order SLIM (HOSLIM)

- Despite effectiveness, the expressiveness of SLIM can be limited by its modeling of pairwise item relations only, since it overlooks the possible higher-order relations, such as multiple items belong the same group, share the same attributes, co-occur frequently, and so on.

- HOSLIM extends SLIM to capture higher-order item relations. In particular, they first apply frequent itemset mining algorithm to identify itemsets that are frequently co-interacted by users; they then extend SLIM to jointly learn the item-item similarities and itemset-item similarity, which can capture higher-order item relations. Specifically, the predictive model of HOSLIM is as follows,

$$\hat{y}_{ui} = r_u^T s_i + r_u'^T s_i'$$

where $r_u \in R^I$ denotes the interaction vector of u on items, and $r_u' \in R^{I'}$ denotes the interaction vector of u on itemsets (I' itemsets in total), where each entry r'_{ui} denotes whether u has interacted with all items in itemset i'. Vectors s_i and s_i' are model parameters to learn, where $s_i \in R^I$ denotes the similarity vector of i on items, and $s_i' \in R^{I'}$ denotes the similarity vector of i on itemsets

- The objective function of HOSLIM is similar to that of SLIM with additional constraints on the itemset similarity matrix as under

$$L_{SLIM} = 1/2 \sum_{u=1}^{U} \sum_{i=1}^{I} (r_{ui} - \hat{y}_{ui})^2 + \lambda_1(||S||_1 + ||S'||_1) + \lambda_2(||S||_2 + ||S'||_2)$$

FISM and NIAS Methods

- FISM is another mainstream in learning-based ICF, instead of directly learning the whole item similarity matrix which can be very space and time consuming, it applies a low-rank latent factors on the item similarity matrix. The predictive model of FISM is formulated as,

$$\hat{y}_{ui} = \frac{1}{(|R|_u^+ - 1)^\alpha} \sum_{j \in R_u^+ / i} r_{uj} p_i^T q_j$$

p_i and q_j are the latent features for target item i and historical item j. As can be seen, the item similarity score s_{ij} can be expressed as the inner product between p_i and q_j . Hyper-parameter α controls the normalization on users of different history length

- NAIS applies a dynamic weighting strategy on second-order item relations by considering that the historical items of u should have different contributions on the prediction. An attention network is then employed to learn the varying weights of item-item relations based on item embeddings. The predictive model of NAIS is formulated as,

$$\hat{y}_{ui} = \sum_{j \in R_u^+ / i} \alpha_{ij} p_i^T q_j$$

where α_{ij} denotes the attentive weight of similarity s_{ij} in contributing to the final prediction. In NAIS, α_{ij} is parameterized as a neural network's output with item embeddings p_i and q_j as input.