Learning Tree-based Deep Model for Recommender Systems

Han Zhu, Pengye Zhang, Guozheng Li, Jie He, Han Li, Kun Gai
Alibaba Group
{zhuhan.zh,pengye.zpy,guozheng.lgz,jay.hj,lihan.lh,jingshi.gk}@alibaba-inc.com

ABSTRACT

We propose a novel recommendation method based on tree. With user behavior data, the tree based model can capture user interests from coarse to fine, by traversing nodes top down and make decisions whether to pick up each node to user. Compared to traditional model-based methods like matrix factorization (MF), our tree based model does not have to fetch and estimate each item in the entire set. Instead, candidates are drawn from subsets corresponding to user's high-level interests, which is defined by the tree structure. Meanwhile, finding candidates from the entire corpus brings more novelty than content-based approaches like item-based collaborative filtering. Moreover, in this paper, we show that the tree structure can also act to refine user interests distribution, to benefit both training and prediction. The experimental results in both open dataset and Taobao display advertising dataset indicate that the proposed method outperforms existing methods.

CCS CONCEPTS

• **Computing methodologies** → **Classification and regression trees**; **Neural networks**; • **Information systems** → **Recommender systems**;

KEYWORDS

Tree-based Learning, Recommender Systems, Implicit Feedback

1 INTRODUCTION

Recommendation has been widely used by various kinds of content providers. Personalized recommendation method, based on the intuition that user interests can be inferred from its historical behavior, or other users with similar preference, has been proven to be effective in YouTube [6] and Amazon [17]. Taobao, as the world's largest online retail platform, also has strong demands for efficient recommender systems to help users find what they want precisely. In this paper, we propose a novel and universal tree-based deep model to solve the personalized recommendation problem in enormous corpus with implicit user feedback data, which is a common problem for many online content providers like E-commerce sites, social media and online news applications.

Designing such a recommendation model has many challenges. First, in the scenario of enormous corpus, some well performed recommendation algorithms could fail when making prediction among the entire corpus. The computational magnitude of some model-based methods like MF to compute the preference over each item for each user is unacceptable. Second, to hold high concurrency and low latency online serving system, the running time of recommendation models to make prediction for each single user is limited. Third, besides preciseness, the novelty of recommended entries should also be responsible for user experience, where homogeneous results are not expected.

In the literatures of collaborative filtering for recommender systems, model-based methods are an active topic of research. Models such as matrix factorization [11, 15, 22] try to decompose pairwise user-item preferences (e.g., ratings) into user and item factors, then recommend to each user its most preferred items. In some real-world scenarios that have no explicit preference but only implicit user feedback data (e.g., user behaviors like clicks or purchases), Bayesian Personalized Ranking [21] gives a solution that formulates the preference in triplets with partial order, and applies it to MF models. Recent work [10] further proposes a neural network based collaborative filtering framework, where a neural network instead of inner product is used to model the relation between user and item latent factors. However, this kind of MF like methods need to compute the preference score of all items when making prediction for a specific user, which makes it very difficult to hold real-time online serving in high concurrency systems with large-scale corpus.

To reduce the online serving latency and handle enormous corpus, content-based collaborative filtering methods are widely deployed in industry [17]. As a representative method in collaborative filtering family, item-based collaborative filtering [23] (item-CF) can recommend from very large corpus with low latency, depending on the pre-calculated similarity between item pairs and using user's historical behaviors as triggers to recall those most similar items. However, the rule-based way that recommending similar items according to historical behavior imposes restriction on the scope of candidate set, i.e., not all items but only the items triggered by user behavior are probably to be ultimately recommended. This intuition severely limits the novelty of recalled results, and stops the recommender system from jumping out of historical behavior to explore potential user interests.

Inspired by the immense progress in deep learning and natural language processing, a series of works have emerged from new perspectives [25]. YouTube introduces its recommender system [6], where the recommendation is posed as an extreme multiclass classification, and sampled softmax [13] is used to reduce computational complexity. In this way, the model can learn both user and item embeddings with user behaviors as input. Such user embedding can response to user interests' evolution instantaneously, and the efficiency of online serving could also be ensured by hashing [24] or quantization [14] indices. Nevertheless, the approximate kNN search manner limits the similarity metric between user and item to inner product or Euclidean distance of their embeddings, which amounts to limit the complexity and capacity of the last mile model that used to distinguish whether user-item pairs are related. Furthermore, as one of the most important transformed features in recommender system [4], the cross-product transformation between user's historical behaviors and candidate items can not take efforts in this kind of system design.

Weighing the pros and cons, we propose a novel tree-based deep model to solve recommendation problem in online systems with very large-scale corpus. Tree and tree-based methods are researched in multiclass classification problem [1, 2, 5], but researchers seldom set foot in the context of recommender systems using tree-based methods. As a kind of efficient index structure, it's a proper way to use tree to organize items, thereby enabling low latency retrieval from entire corpus. And the tree structure itself can also be learnt to adapt recommendation problem. The tree-based model opens a new perspective of view in large-scale recommendation, which is not exclusive for Taobao's E-commerce scenario, but a universal solution for large-scale recommender systems. Tree-based approach has the following advantages:

- As a kind of widely used index structure, searching in a tree is highly efficient. It usually takes only logarithmic time complexity to find the results, which is sufficient to retrieve from enormous corpus. This kind of recommendation approach surpasses item-CF like methods which are based on explicit historical behaviors and similarity calculation rules, by exploring user interests from the entire corpus using a trained model.
- Analogy to a tree, items or products in the corpus can be organized in a hierarchical category system naturally. When traversing from root to leaf in such a tree, user interests are narrowing down from coarse to fine, and can also propagate among the same level nodes. Besides training the model, the tree structure can also be learnt towards optimal organization of items for more effective retrieval, which in turn benefits the model training. The tree based approach allows jointly training of recommendation model and retrieval tree structure.
- When retrieving in a tree, usually only limited number of internal nodes needed to be traversed, depending on the final top-k number required. Compared to approximate kNN search manner where the similarity metric is limited, tree-based approach makes it possible to use richer feature types and more effective models when discriminating each node.

The remainder of this paper is organized as follow: In Section 2, we'll introduce the system architecture of Taobao display advertising to show the position of recommendation. Section 3 will give a detailed introduction and formalization of the proposed tree-based deep model. And the following Section 4 will describe how the tree-based model serve online. Experimental results on open dataset and Taobao advertising dataset are shown in Section 5. At last, Section 6 gives our work a conclusion.

2 SYSTEM ARCHITECTURE

In this section, we introduce the architecture of Taobao display advertising recommendation system as Figure 1, to help understand how the proposed algorithm works in each page view request.

After receiving page view request from a user, the system uses user features, context features and item features as input to generate a relatively much smaller set (usually hundreds) of candidate items in the matching server. The tree based recommendation model takes effort in this stage and shrinks the size of candidate set by several orders of magnitude.

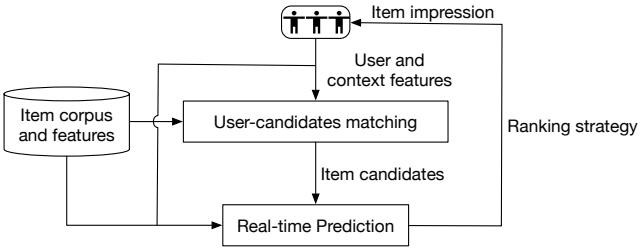

Figure 1: The system architecture of Taobao display advertising recommendation system.

With hundreds of candidate items, the real-time prediction server uses a more elaborate but also more time consuming model [26] to predict indicators like click-through rate or conversion rate. And after ranking by strategy, several items are ultimately impressed to user.

As aforementioned, the proposed recommendation model aims to construct a candidate set with hundreds of items. This stage is essential and also difficult. Whether the user is interested in the generated candidates gives an upper bound of the impression quality. And the enormous corpus size limits the model complexity. In one word, how to draw high quality candidates with lightweight model is the problem.

3 TREE-BASED DEEP MODEL

In this part, we first formulate the tree structure used in our tree-based model and the training objective function to give an overall conception. Secondly, we describe the deep neural network architecture. At last, we introduce the detailed training strategies of the proposed tree-based deep recommendation model.

3.1 Tree For Recommendation

A recommendation tree T consists of a set of nodes N and a set of edges E, where $N = \{n_1, n_2, \cdots n_{|N|}\}$ represents $|N|$ individual non-leaf or leaf nodes, and $E = \{(p_1, c_1), (p_2, c_2), \cdots, (p_{|E|}, c_{|E|})\}$ in which each pair (p_i, c_i) means the edge that connects parent node n_{p_i} and child node n_{c_i}. Each node in N except the root node has one parent and an arbitrary number of children. Specifically, each item c_i in the corpus C corresponds to a leaf node in the tree T. Without loss of generality, we suppose that node n_1 is the root node, and the set of leaf nodes is $L = \{n_i | i = |N| - |C| + 1, \cdots, |N|\}$, i.e., the set of $|C|$ nodes with largest indices. An example tree T is illustrated in Figure 2, in which each circle represents a node and the number of node is its index in tree. The tree has 8 leaf nodes in total, each of which corresponds to an item in the corpus. It's worth to mention that though the given example is a complete binary tree, we don't impose this restriction on the type of the tree in the following of this paper.

Recommending items for a user in a recommendation tree T thus is to find a set of (e.g., size k) leaf nodes for the user. Suppose n_i is a leaf node, i.e., an item in the corpus that user u is interested in. We derive it as $\hat{y}_u(n_i) = 1$, where $\hat{y}_u(\cdot)$ is a function to tell the ground truth whether user u is interested in a node or not. When retrieving data in a general tree-based index structure, the nodes

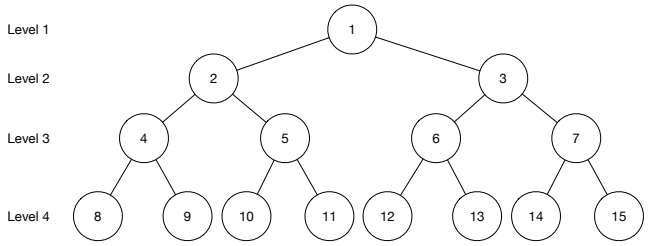

Figure 2: An example of the recommendation tree that has 4 levels and 15 nodes, 8 in which are leaf nodes.

are usually traversed in a layer-wise and top-down sequence. In our tree-based recommendation approach, the tree not only acts as an index, but also organizes the corpus in hierarchy. Therefore, considering the retrieving process and the hierarchy information, the objective function is proposed as

$$P\Big(y_u\big(l_1(n_i)\big) = 1, \cdots, y_u\big(l_m(n_i)\big) = 1 \Big| l_1(n_i), \cdots, l_m(n_i), u\Big) = \prod_{j=1}^{m} P\Big(y_u\big(l_j(n_i)\big) = 1 \Big| y_u\big(l_1(n_i)\big) = 1, \cdots, y_u\big(l_{j-1}(n_i)\big) = 1, l_j(n_i), u\Big), \tag{1}$$

where $l_j(n_i)$ is the ancestor node of n_i in level j, m is the level of n_i in the tree, and $l_m(n_i)$ is n_i itself. Objective 1 is, for the node n_i that user is interested in, we maximize the joint probability of user is interested in each node that lies in n_i's path to the root node.

The formulation of the joint probability is closely related to tree's indexing mechanism and the corpus's hierarchical structure. In tree as a index like B-Tree or binary search tree, one node in each level is traversed to find the indexed record. In our scenario, for the sake of that the desired leaf nodes can be recalled, traversing each of its ancestor node is the prerequisite, i.e., the model should predicate all ancestor nodes as related ones to the user. Like ontology trees in natural language where there exists belongingness between low-level and high-level semantic concepts, nodes' concepts in recommendation tree also have similarity or dependency between parents and children. High-level nodes represent those broad categories in a sense and low-level or leaf nodes contain fine-grained ones. Therefore, the joint optimization is appropriate for both retrieval and the intention to model user interests.

In the recommendation tree, each node has a corresponding semantic concept from the corpus. For example, each leaf node is an individual item, and a non-leaf node means the unity of all it's direct/indirect children. Without loss of generality, each node can be assigned to a feature vector, which can represent the conditions related to its ancestors in Objective 1. Therefore, Objective 1 is refined to the following form:

$$\prod_{j=1}^{m} P\Big(y_u\big(l_j(n_i)\big) = 1 \Big| l_j(n_i), u\Big). \tag{2}$$

Given the definition of recommendation tree, the detailed retrieval algorithm is described in Algorithm 1.

Algorithm 1: Retrieval Algorithm

Input: User u and its features, the recommendation tree T, the desired candidate number k
Output: The set of recommended leaf nodes

```
A = ∅, Q = {n_1};
repeat
    Calculate the probability P(y_u(n) = 1|n, u) that u will be interested in the node n for each node n ∈ Q;
    Sort nodes in Q in descending order of P(y_u(n) = 1|n, u) and derive the set of top k nodes as I;
    Insert the leaf nodes in I into A;
    Clear Q and insert all the non-leaf children of nodes in I into Q;
until |Q| == 0;
Return the top k items in set A, according to P(y_u(n) = 1|n, u);
```

3.2 The Deep Model

In the following part, we introduce the deep model we use. As illustrated in Figure 3, the entire model can be divided into two parts, the deep neural network and the tree. Inspired by the language model [18] and a click-through rate prediction work [26], We learn high dimensional embeddings for each node in the tree. To exploit user behavior that contains timestamp information, we designed the block-wise input layer to discriminate behaviors that lie in different time windows and types. For example, the historical behavior can be grouped by days or weeks, and item embeddings in each group is averaged. Of course, other methods like recurrent neural network (RNN) can be used to modeling user behavior as a sequence [7]. However, the computational complexity of RNN is much larger than a fully connected layer, which could bring too much overhead in high concurrency online system.

The embeddings of tree nodes and the tree structure itself are also parts of the model. To maximize Objective 2, the probability $P(y_u(n_i) = 1|n_i, u)$ of each node n_i in user's interested leaf node's path to the root is maximize. Of course, negative samples are essential and we'll introduce the negative sampling in the next section. But here, one of the key points is the binary classifier illustrated in Figure 3. The output after three fully connected layers is regarded as the vector representation of user, along with the node embedding, a binary classifier is trained to estimate whether the user is interested in the node. It's worth to emphasize that the binary classifier can be a simple sigmoid function that uses the inner product of user vector and node embedding as input, or other more complicate and effective ones like another neural network. The choice of binary classifier should weighing between capacity and complexity, and we analyse the results of different classifiers in Section 5.

3.3 Model Training

As described in Section 3.1, the proposed tree-based deep model should guarantee that the right leaf nodes can be retrieved given a user vector. In this part, we introduce how to train the model.

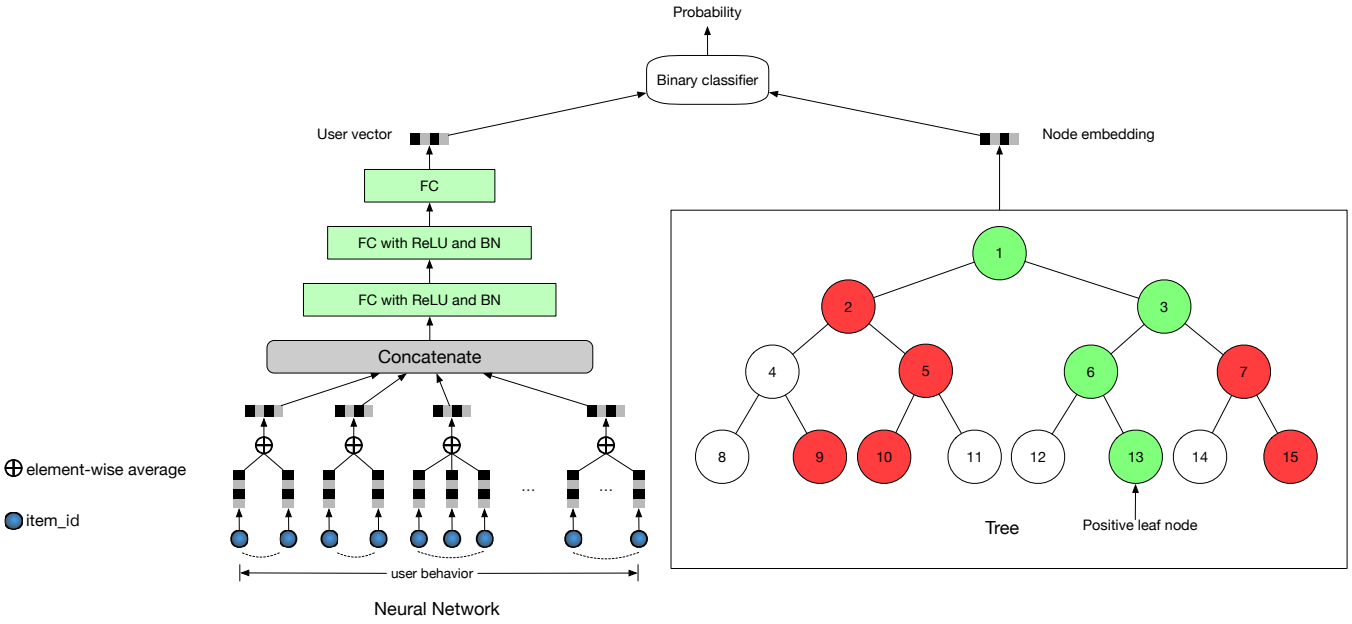

Figure 3: The tree based deep model architecture. The neural network part receives the embedded user behavior as input and outputs the user vector after three fully connected layers with ReLU [20] as activation. The first two fully connected layers are following by a batch normalization [12]. User behaviors are placed in different blocks according to the action time and the behavior type, to distinguish their contributions to user future interests. Usually about tens of blocks are used in practice. In training or online serving, the binary classifier takes the user vector along with one node's embedding as input to train or make prediction.

Sampling. To maximize Objective 2, the positive samples for model training can be directly found. As that most user feedback in Taobao are implicit ones like user clicks, we use these implicit feedback as positive samples. Specifically, user behaviors including clicks, purchases and adding items into the shopping cart are treated as positive samples. Nodes that lie in the positive sample's path to root are also positive.

Some careful readers may have found that the proposed tree method is similar to hierarchical softmax [19]. But they are actually quite different in several aspects. First of all, the target tasks are different. Hierarchical softmax in language model tries to solve the next-one prediction problem in a sequence, further to learn word embeddings, while the proposed tree-based approach aims to predict comprehensive user interests. Second, an either-or discrimination is made between the two children of each node according to the node's embedding in hierarchical softmax, which is unreasonable in recommendation scenario, because user may be interested in all the children nodes which have similar concepts. Third, our recommendation tree is not limited to a binary one, where the 0/1 encoding approach is not appropriate.

For negative sampling in implicit feedback recommendation problem, a common way is to random sample from the dataset[6]. Here we use layer-wise random selection of the nodes to generate negative samples. As the pyramid like tree structure, a different number of negative nodes in each layer, along with the positive nodes together make up the bunch of samples for each user behavior as the target. Figure 3 gives a sampling example. Suppose that n_{13} is the target leaf node, then all the nodes in green are labeled as positive and random selected nodes in red are labeled as negative. In practice, the root node is always a positive one that can be omitted.

Loss function. For each user behavior in the training set, user actions before the behavior time can be organized as the neural network input to produce a user vector, and node samples can be drawn from the tree to get node embeddings. Given a user vector u, the cross-entropy loss is minimized for each sampled nodes. The entire loss function is derived as

$$-\sum_{u}\sum_{n}\hat{y}_u(n)\log P\big(y_u(n)=1|n,u\big)+\big(1-\hat{y}_u(n)\big)\log P\big(y_u(n)=0|n,u\big), \tag{3}$$

where $P\big(y_u(n)=1|n,u\big)$ is the output probability of the binary classifier.

4 ONLINE SERVING

Figure 4 illustrated the online serving system of the proposed method. User vector calculation and item retrieval are split into two asynchronous stages. Each user behavior including clicks, buys and adding item into shopping cart will strike the real-time feature server to calculate a new user vector using the trained neural network. And once receiving page view request, the user targeting server will use the pre-calculated user vector to retrieve candidates

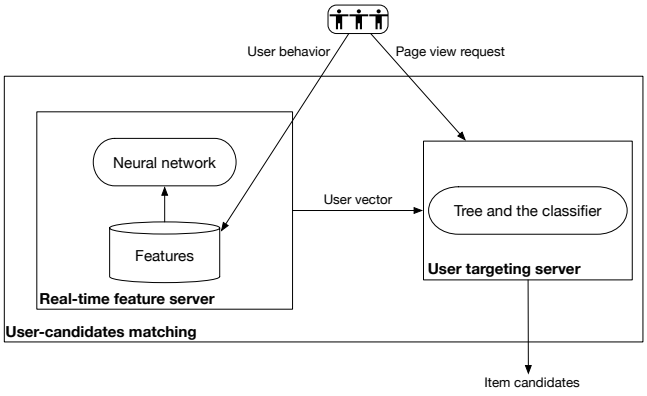

Figure 4: The online serving system of the tree-based model, where the main feature is asynchronous calculation of user vector. Each user behavior will strike the real-time feature server to calculate the user vector, which can be directly used by the user targeting server when receiving a page view request.

from the tree. As described in Algorithm 1, the retrieval is layer-wise and the trained binary classifier is used to calculate the probability that whether a node is preferred given the user vector.

The main consideration of asynchronous design is to leave more time for the retrieval stage. Section 5 will show that a more precise but also more time consuming binary classifier could bring significant gain in performance.

5 EXPERIMENTAL STUDY

We study the performance of the proposed tree-based model in this section. Experimental results in open dataset and Taobao advertising dataset are presented. In the experiments, we compare the proposed method to other existing methods to show the effectiveness of the model, and empirical study results show how the tree-based model works.

5.1 Datasets

The main purpose of the proposed tree-based model is to recommend in enormous corpus according to user behavior. Thus, the experimental studies are conducted in two large-scale datasets with timestamps: 1) users' movie viewing data from MovieLens [9]; 2) a user-item behavior dataset from Taobao. In more details:

MovieLens-20M: It contains user-movie ratings with timestamps in this dataset. As we deal with implicit feedback problem, the ratings are binarized by keeping the ratings of four or higher, which is a common way in other works [7, 16]. Besides, only the users who have watched at least 10 movies are kept. To create training, validation and testing sets, we random sample 1,000 users as testing set and another 1,000 users as validation set, while the rest users make up the training set. For validation and testing set, the first half of user-movie views along the timeline are regarded as known behavior to predict the latter half.

UserBehavior: This dataset is a subset of Taobao user behavior data. We random select 1 million users who have at least 10 behaviors including item view, purchase and adding item to shopping cart during November 25, 2017 to December 03, 2017. The data is organized in a very similar form to MovieLens-20M, i.e., a user-item behavior is consist of user ID, item ID, item's category ID, behavior type and timestamp. As we do in MovieLens-20M, 10,000 users are random selected as testing set and another random selected 10,000 users are validation set. The ground truths are also the latter half of the behaviors along the timeline for testing and validation sets.

Table 1 summarized the major dimensions of the above two datasets after preprocessing.

	MovieLens-20M	UserBehavior
# of users	129,797	1,000,000
# of items	20,709	4,023,451
# of categories	20	9,378
# of records	9,939,873	100,934,102

Table 1: Dimensions of the datasets after preprocessing. One record is a user-item pair with timestamp that represents a implicit user feedback.

5.2 Metrics and Comparison Methods

To evaluate the effectiveness of different candidate generation methods, we use Precision@M, Recall@M and F-Measure@M metrics. Derive the recommended set of items for a user u as $\mathcal{P}_u$ ($|\mathcal{P}_u| = M$) and the user's ground truth set as $\mathcal{G}_u$. Precision@M is

$$Precision@M(u) = \frac{|\mathcal{P}_u \cap \mathcal{G}_u|}{M}, \tag{4}$$

Recall@M is

$$Recall@M(u) = \frac{|\mathcal{P}_u \cap \mathcal{G}_u|}{|\mathcal{G}_u|}, \tag{5}$$

and F-Measure@M is

$$F_1@M(u) = \frac{2 * Precision@M(u) * Recall@M(u)}{Precision@M(u) + Recall@M(u)}. \tag{6}$$

As we emphasize, recommendation results' novelty is responsible for user experience. Thus, metric to measure novelty is used to compare different methods.

Existing work [3] gives several approaches to measure the novelty of recommended list of items. Following one of its definition, the Novelty@M is defined as

$$Novelty@M(u) = \frac{|\mathcal{P}_u \setminus \mathcal{S}_u|}{M}, \tag{7}$$

where $\mathcal{S}_u$ is the set of items that have been consumed by user u before recommendation.

User average of the above four metrics in testing set are used to compare the following methods:

- **BPR**[21]. We use its matrix factorization form for implicit feedback recommendation. The objective function that is similar to AUC (area under the ROC curve) is optimized. We use the implementation of BPR provided by [8]. Like other

MF methods, BPR requires that the users in testing or validation set also have feedback in training set. Therefor, we add the first half of user-item interactions along the timeline in testing and validation set into the training set in both MovieLens and UserBehavior dataset.

- **Item-kNN**[23]. Item-based collaborative filtering is one of the most widely used personalized recommendation method in production with large-scale corpus [17]. It's also one of the major candidate generation approaches in Taobao. We use the implementation of item-kNN provided by Alibaba machine learning platform.
- **RAC**[6] (recommendation as classification) is the deep recommendation approach proposed by YouTube. Recommendation is posed as an extreme multiclass classification, and sampled softmax [13] is used to reduce computational complexity. We implement RAC in our deep learning platform with the same network architecture and features with the proposed tree-based model. Exact kNN search between user and item embeddings in inner product space is adopted when making recommendation.
- **TDM** (tree-based deep model) is our proposed method. The inner product of user vector and node embedding with sigmoid activation acts as the binary classifier in Figure 3 for fairness. The tree is built in an intuitive way: items are ranked by their categories (items with more than one category is assigned to a random selected one) and halved to two equal parts recursively, which could construct an almost-complete binary tree top-down.

For BPR and item-kNN, we tune several most important hyperparameters based on the validation set, i.e., the number of factors and iterations in BPR, the number of neighbors in item-kNN. For RAC and TDM, the node and item embeddings' dimension is set to 24, and the hidden unit numbers of the three fully connected layers are 128, 64 and 24.

5.3 Comparison Results

The comparison results of different methods are shown in Table 2. Each metric is averaged across all the users in testing set, and the presented values are the average across five different runs for methods with variance. The results indicate that the proposed TDM approach outperforms all the baselines in both datasets on most of the metrics.

In MovieLens-20M, TDM gets significantly better results than the widely used collaborative filtering method item-kNN, especially on recall and novelty metrics. In UserBehavior dataset, the gap between TDM and item-kNN shrinks on precision and recall. However, the proportion of items in the recommendation results that user has already consumed is much lower in TDM (1.59% compared to 2.94%). And the following Section 5.4 will show that TDM performances much better in discovering user's potential interests.

Compared to the other widely deployed deep learning and vector kNN search based method RAC, TDM also gets better results on most of the metrics. It's worth to mention that when implementing RAC, the exact kNN search in inner product space is adopted, which is usually replaced by approximate ones that could cause performance reduction.

5.4 Empirical Analysis

Recommendation Novelty. Recommending novel items for users to avoid homogeneous results is essential for good user experience. On the one hand, novelty is to recommend unconsumed items for user; on the other hand, the recommendation should also be useful. Results in Table 2 has shown that the recommendation results of TDM has little interaction with users' consumed item set, especially in UserBehavior dataset. However, the effectiveness of those novel recommendations also should be verified. As a user may consume a same item many times, we do experiments by removing those consumed items in result set to verify the novelty. Results in Table 3 indicates that TDM significantly outperforms item-kNN.

When consumed items are removed from recommended set, all metrics drop compared to the results in Table 2. The reason is that user historical behaviors in e-commerce scenario are very strong user interests and relatively easier to predict. Another meaningful observation is that the performance drop rate of TDM is smaller than item-kNN, which proves the advantage of making prediction from the entire corpus rather than the subset that triggered by user's historical behavior.

Variants of TDM. To comprehend the proposed tree-based model itself, we derive and evaluate several variants of TDM:

- **TDM-N.** In Section 3.3 we describe how the proposed model is trained and its differences with hierarchical softmax [19]. An important point is their different ways in negative sampling. Here we test the variant TDM-N that samples a positive node's neighbor as a negative sample. As we use binary trees in our experiments, the ratio of positive and negative samples is near 1:1.
- **TDM-D.** When making top k recommendation for users, we use Algorithm 1 to retrieve leaf nodes in the tree. TDM-D aims to test how important the tree structure is in the proposed method. It directly selects k nodes among all the leaf nodes with largest probability $P(y(n) = 1|n, u)$, and the training stage is the same with original TDM.
- **TDM-C.** As mentioned that one important advantage of tree-based model is the similarity metric is not limited in inner product space or Euclidean space. In original TDM, the inner product between user vector and node embedding that goes through a sigmoid function makes up the binary classifier, where the similarity is still measured in inner product space. TDM-C moves the node embedding in Figure 3 to the bottom layer of the neural network, along with the user behavior feature consists of the network input. The binary classifier thus is changed to a 3-layers neural network with two classes softmax.

The experimental results in MovieLens-20M dataset of the above three variants are in Table 4. The failure of TDM-D indicates that the tree structure is essential and significantly improves the recommendation quality. TDM-C outperforms all the testing methods, which proves the advantage of tree-based recommendation that can use more flexible similarity metrics. The results of TDM-N shows that a hierarchical softmax like method can not fit in recommendation problem very well, as we analyzed in Section 3.3.

Method	MovieLens-20M (@10)				UserBehavior (@200)			
	Precision	Recall	F-Measure	Novelty	Precision	Recall	F-Measure	Novelty
BPR	0.0810	0.0509	0.0502	0.6256	0.0076	0.0384	0.0117	0.9761
Item-kNN	0.0825	0.0566	0.0529	0.5946	0.0094	0.0493	0.0141	0.9706
RAC	0.1187	0.0871	0.0796	0.7138	0.0087	0.0502	0.0133	**0.9863**
TDM	**0.1220**	**0.0918**	**0.0823**	**0.7278**	**0.0096**	**0.0549**	**0.0159**	0.9841

Table 2: The comparison results of different methods in MovieLens-20M and UserBehavior dataset. According to the different corpus size, metrics are evaluated @10 in MovieLens-20 and @200 in UserBehavior. The proposed TDM outperforms all the baselines in both datasets on most of the metrics.

Role of the tree. Tree is the key component of the proposed tree-based model. It not only acts as an index when retrieving, but also abstracts user interests coarse-to-fine. Fine-grained recommendation is much more difficult than coarse-grained. For example, it's not hard to decide whether to recommend shoes to a young women in basis of her browsing history, but it's not so easy to directly pick out a pair of snow boots from the entire corpus for a recommendation model. The tree based abstraction of user interests makes it possible to solve the problem progressively, from easy to difficult. Figure 5 illustrates the layer-wise precision and recall metrics in MovieLens-20M dataset. An observation is that as the node number in a level goes up, the precision and recall go down correspondingly, which shows that the tree divides the recommendation problem into smaller subparts and solve them one by one.

Retrieval efficiency. One of the barriers in recommender system for large-scale corpus is the limitation of serving latency, and it's an important motivation of proposing our tree-based model. Here we have experiments to show the retrieval efficiency. The tree structure we use is a incomplete binary tree with max level of 40 and about 2 millions of leaf nodes. The retrieval process is implemented in C++ and runs on CPU without parallel. Results in Table 5 are the average retrieval time of 10, 000 different users. The results indicate that the retrieval time is near linear-scale w.r.t. the quantity of finally recalled items. For the online serving, several milliseconds' latency can fulfill the demands.

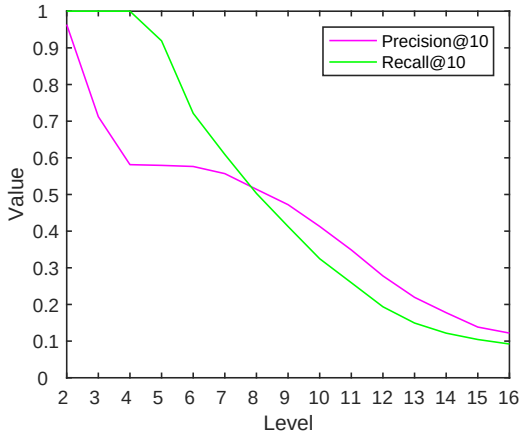

Figure 5: The results of layer-wise Precision@10 and Recall@10 in MovieLens-20M dataset. The ground truths in testing set are traced back to each node's parents, till the root node.

Method	Precision@200	Recall@200	F-Measure@200
Item-kNN	0.0051	0.0275	0.0076
TDM	**0.0069**	**0.0375**	**0.0102**

Table 3: Results of item-kNN and TDM in UserBehavior dataset by excluding consumed items from the recommendation results.

Method	Precision@10	Recall@10	F-Measure@10
TDM-N	0.0785	0.0523	0.0519
TDM-D	0.0806	0.0597	0.0543
TDM-C	**0.1334**	**0.1009**	**0.0897**

Table 4: Results of three variants of TDM in MovieLens-20M dataset. TDM-C significantly outperforms the other two variants, including the original TDM. TDM-D that ignores the tree structure when making prediction gets poor performance. TDM-N that uses neighbor negative sampling also underperforms, even worse than item-kNN.

Method	$k = 100$	$k = 200$	$k = 400$
TDM	1.49 ms	2.87 ms	5.79 ms

Table 5: Retrieval efficiency w.r.t. different numbers k of finally recalled items. The dimension of user vector and node embedding are both 24.

6 CONCLUSION

We figure out a number of challenges for recommending in large-scale corpus, including the efficiency problem and how to present novel and useful items to user. A tree-based deep approach is proposed, where the neural network can infer user interests according to its historical behavior, along with the tree structure to model the interests coarse-to-fine. We come up with an objective function for the recommendation tree and introduce in detail how the model is trained. Comprehensive experimental results prove the effectiveness of the tree-based model, both in recommendation accuracy and novelty. Empirical analysis showcases how and why the proposed model works. In Taobao display advertising platform, the

tree-based recommender system has been deployed in production, which has improved both business benefits and user experiences.

REFERENCES

[1] Samy Bengio, Jason Weston, and David Grangier. 2010. Label embedding trees for large multi-class tasks. In *International Conference on Neural Information Processing Systems*. 163–171.

[2] Alina Beygelzimer, John Langford, and Pradeep Ravikumar. 2007. Multiclass classification with filter trees. *Gynecologic Oncology* 105, 2 (2007), 312–320.

[3] Pablo Castells, SaÃžl Vargas, and Jun Wang. 2011. Novelty and Diversity Metrics for Recommender Systems: Choice, Discovery and Relevance. *In Proceedings of International Workshop on Diversity in Document Retrieval (DDR* (2011), 29–37.

[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 7–10.

[5] Anna Choromanska and John Langford. 2015. Logarithmic time online multiclass prediction. *Computer Science* (2015), 55–63.

[6] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *ACM Conference on Recommender Systems*. 191–198.

[7] Robin Devooght and Hugues Bersini. 2016. Collaborative Filtering with Recurrent Neural Networks. (2016).

[8] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. MyMediaLite: A free recommender system library. In *Proceedings of the fifth ACM conference on Recommender systems*. ACM, 305–308.

[9] F. Maxwell Harper and Joseph A. Konstan. 2015. *The MovieLens Datasets: History and Context.* ACM. 19 pages.

[10] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 173–182.

[11] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 549–558.

[12] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. (2015), 448–456.

[13] SÃlbastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On Using Very Large Target Vocabulary for Neural Machine Translation. *Computer Science* (2014).

[14] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2011. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence* 33, 1 (2011), 117–128.

[15] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

[16] Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M. Blei. 2016. Factorization Meets the Item Embedding:Regularizing Matrix Factorization with Item Co-occurrence. In *ACM Conference on Recommender Systems*. 59–66.

[17] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 7, 1 (2003), 76–80.

[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *International Conference on Neural Information Processing Systems*. 3111–3119.

[19] Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. *Aistats* (2005).

[20] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *International Conference on International Conference on Machine Learning*. 807–814.

[21] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

[22] Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic Matrix Factorization. In *International Conference on Neural Information Processing Systems*. 1257–1264.

[23] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *International Conference on World Wide Web*. 285–295.

[24] J. Weston, A. Makadia, and H. Yee. 2013. Label partitioning for sublinear ranking. In *International Conference on Machine Learning*. 181–189.

[25] Shuai Zhang, Lina Yao, and Aixin Sun. 2017. Deep Learning based Recommender System: A Survey and New Perspectives. (2017).

[26] Guorui Zhou, Chengru Song, Xiaoqiang Zhu, Xiao Ma, Yanghui Yan, Xingya Dai, Han Zhu, Junqi Jin, Han Li, and Kun Gai. 2017. Deep Interest Network for Click-Through Rate Prediction. (2017).