Adapting to Drifting Preferences in Recommendation

Fei Mi, Boi Faltings
Artificial Intelligence Lab, EPFL
Lausanne, Switzerland
firstname.lastname@epfl.ch

Abstract

Machine learning is often used to acquire knowledge in domains that undergo frequent changes, such as networks, social media, or commercial markets. These frequent changes pose a challenge to most machine learning methods as they have difficulty adapting. In this paper, we consider a forum content recommender system for massive open online courses (MOOCs) as an example where recommendations have to adapt to new items and evolving user preferences. We formalize the recommendation problem as a sequence prediction problem and compare different recommendation methods, including a new method called context tree (CT), which can be efficiently learned and applied to online sequential recommendation task. The results show that the CT recommender performs better than other methods because of better adaptation to changes in the domain.

1 Introduction

With the increased availability of data, machine learning has become the method of choice for knowledge acquisition in intelligent systems and various applications. However, data and the knowledge derived from it have a timeliness, such that in a dynamic environment not all of the knowledge acquired in the past remains valid. Therefore, machine learning models should acquire new knowledge incrementally and adapt to the dynamic environments. Many statistical machine learning techniques interpolate between input data and thus their models can adapt only slowly to new situations. As an example, drifting user interests and preferences [1, 6] are important in building personal assistance systems, such as recommendation system for social networks or for news websites where recommendations need be adaptive to drifting trends rather than recommending obsolete or well-known information. Traditional techniques, such as collaborative filtering, only adapt slowly in this case as they build an increasingly complex model of users and items. Therefore, when a new item is superseded by a newer version or a new preference pattern is reached, it takes time for recommendations to adapt.

Our perspective sees recommendation problem as a dynamic and sequential machine learning problem, and it can be understood as the task of predicting the next item in a sequence of items consumed by that user. We use algorithms for sequential recommendation based on variable-order Markov models. More specifically, we use the context tree method developed by [7, 8], and applied to news recommendation by [3, 4]. This structure elegantly combines old knowledge with new knowledge so that it identifies different contexts when they are valid and appropriate. In experiments, it is compared with various sequential and non-sequential methods. We show that both old knowledge and new patterns can be captured fast and accurately through context activation using CT, and that this is why it is particularly strong at adapting to drifting user preferences and performs much better. We thus believe that algorithms for sequential prediction, such as CT, have great potential to improve the performance of intelligent systems in domains where adaptation is important.

2 Context Tree (CT) for Recommendation

Context tree is a space efficient structure to keep track of the history in a variable-order Markov chain so that a data structure is built only for sequences that actually occur. Our CT recommender is an effective generalization of variable-order Markov models to full on-line Bayesian estimation as a
Dirichlet prior is added for each expert conditioned on the context. The resulting construction uses a recursion and can be updated efficiently, and it allows the model to make predictions using more complex contexts as more data is acquired.

2.1 The Structure of Context Tree
In CT, a sequence \( s = \langle n_1, \ldots, n_l \rangle \) is an ordered list of items \( n_i \in \mathbb{N} \) consumed by a user. The sequence of items viewed until time \( t \) is denoted as \( s_t \) and the set of all possible sequences is denoted by \( S \). A context \( \xi : s \) is the set of all possible sequences in \( S \) ending with the suffix \( \xi \). \( \xi \) is the suffix of \( s \) if last elements of \( s \) are equal to \( \xi \). A context tree \( T = (V, E) \) with nodes \( V \) and edges \( E \) is a partition tree over the all contexts of \( S \). Each node \( i \in V \) in the context tree corresponds to a context \( S_i = \{ s \in S : \xi_i \prec s \} \). Initially the context tree \( T \) only contains a root node with the most general context. Every time a new item is consumed, the active leaf node is split into a number of subsets, which then become nodes in the tree. This construction results in a variable-order Markov model. Figure 1 illustrates a simple CT with some sequences over an item set \( \langle n_1, n_2, n_3 \rangle \).

![Figure 1: An example context tree. For the sequence \( s = \langle n_2, n_3, n_1 \rangle \), nodes in red-dashed are activated.](image-url)

2.2 Applying Context Tree to Recommendation
For each context \( S_i \), an expert \( \mu_i \) is associated in order to compute the estimated probability \( P(n_{t+1} | s_t) \) of the next item \( n_{t+1} \) under this context. A user’s browsing history \( s_t \) is matched to the CT and identifies a path of matching nodes (see Figure 1). All the experts associated with these nodes are called active. The set of active experts \( \mathcal{A}(s_t) = \{ \mu_i : \xi_i \prec s_t \} \) is the set of experts \( \mu_i \) associated to contexts \( S_i = \{ s : \xi_i \prec s_t \} \) such that \( \xi_i \) are suffix of \( s_t \). \( \mathcal{A}(s_t) \) is responsible for the prediction for \( s_t \).

2.2.1 Expert Model
The standard way for estimating the probability \( P(n_{t+1} | s_t) \), as proposed by [2], is to use a Dirichlet-multinomial prior for each expert \( \mu_i \). The probability of viewing an item \( x \) depends on the number of times \( \alpha_{xt} \) the item \( x \) has been consumed when the expert is active until time \( t \). The corresponding marginal probability is given in Eq. 1.

\[
P_i(n_{t+1} = x | s_t) = \frac{\alpha_{xt} + \alpha_0}{\sum_{j \in \mathbb{N}} \alpha_{jt} + \alpha_0} \tag{1}
\]

2.2.2 Combining Experts to Prediction
When making recommendation for a sequence \( s_t \), we first identify the set of contexts and active experts that match the sequence. The predictions given by all the active experts are combined by mixing the recommendations given by them:

\[
P(n_{t+1} = x | s_t) = \sum_{i \in \mathcal{A}(s_t)} u_i(s_t) P_i(n_{t+1} = x | s_t) \tag{2}
\]

The mixture coefficient \( u_i(s_t) \) of expert \( \mu_i \) is computed in Eq. 3 using the weight \( w_i \in [0, 1] \). Weight \( w_i \) is the probability that the chosen recommendation stops at node \( i \) given that it can be generated by the first \( i \) experts.
The combined prediction of the first $i$ experts is defined as $q_i$ and it is computed using the following recursion in Eq. 4:

$$q_i = w_i P_i(n_{t+1} = x | s_t) + (1 - w_i)q_{i-1}$$

The weights are updated by taking into account the success of a recommendation. When a user consumes a new item $x$, we update the weights of the active experts corresponding to the suffix ending before $x$ according to the probability $q_i(x)$ of predicting $x$ sequentially via Bayes’ theorem. The weights are updated in closed form in Eq. 5, and a detailed derivation can be found in [2].

$$w_i' = \frac{w_i P_i(n_{t+1} = x | s_t)}{q_i(x)}$$

### 2.3 Adaptation Analysis

Our hypothesis, which is validated in experiments later, is that the CT model can be applied elegantly to domains where adaptation and timeliness are of concern. Two properties of the CT methods are crucial to the goal. First, the model parameter learning process and recommendations generated are online such that the model adapts continuously to a dynamic environment. Second, adaptability can be achieved by the CT structure itself as knowledge is organized and activated by context. New items or paths are recognized in new contexts, whereas old items can still be accessed in their old contexts. This context organization and context matching mechanism help new patterns to be recognized efficiently so as to adapt to changing concepts.

### 2.4 Complexity Analysis

Learning CT uses the recursive updates defined in Eq. 4 and recommendations are generated by weighting the experts’ predictions along the activated path given by Eq. 2. For trees of depth $D$, the time complexity of model learning and prediction for a new observation are both $O(D)$. For input sequence of length $T$, the updating and recommending complexity are $O(M^2)$, where $M = \min(D, T)$. Space complexity in the worst case is exponential to the depth of the tree. However, as we do not generate branches unless the sequence occurs in the input, we achieve a much lower bound determined by the total size of the input. So the space complexity is $O(N)$, where $N$ is the total number of observations as we never generate more than $N$ nodes in the tree. For tasks that involve very long sequences, we can limit the depth $D$ of the CT for space and time efficiency. To further reduce complexity, old branches that have not been activated for long time can be pruned.

### 3 Results and Evaluation

#### 3.1 Problem Description

We focus on the application of recommender systems to the online discussion forums of massive open online courses (MOOCs). The forum viewing data are from three courses offered on Coursera by our university, referred to Course 1, Course 2 and Course 3. Through analyzing the thread viewing pattern we found that there is a sharp trend that fresh forum threads are viewed much more frequently than old ones. It is mainly due to the fact that fresh threads are closely relevant to the current course progress. Moreover, fresh threads can also supersede the contents in some old ones to be viewed. This tendency to view fresh items leads to drifting user preferences.

#### 3.1.1 Comparison of Sequential Methods

We primarily compare CT with sequential methods where the model is updated continuously as items are consumed. Each method recommend top 5 choices each time. Fresh_1 recommends the last 5 updated threads, and Fresh_2 recommends the last 5 created threads. We also consider an online version of MF that is currently the state-of-the-art sequential recommendation method, referred to “online-MF”. The results presented in the table below show the performance of the CT recommender compared with other sequential baseline methods under different settings and evaluation metrics. Succ@5 is the mean average precision (MAP) of predicting the next thread view, and Succ@5Ahead is the MAP of predicting the future thread views within a sequence. Each result tuple contains the performance on the three datasets. We also consider a tail performance
metric, referred to personalized evaluation, where the most popular threads are excluded from recommendations. Separated sequences treat one visit session as one sequence, and combined sequences concatenate all of a student’s visit sessions into one longer sequence. We can see that the CT recommender outperforms all other sequential methods under various settings. We notice that the online-MF method performs much worse compared with the CT recommender. This result shows that matrix factorization, which is based on interpolation over the user-item matrix, is not sensitive enough to rapidly drifting preferences with limited observations. Moreover, simply recommending fresh items even does a better job than online-MF for this task that involves drifting preferences.

<table>
<thead>
<tr>
<th></th>
<th>Non-personalized</th>
<th>Personalized</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Succ@5</td>
<td>Succ@5Ahead</td>
</tr>
<tr>
<td>CT</td>
<td>25, 23, 211%</td>
<td>48, 53, 52%</td>
</tr>
<tr>
<td>online-MF</td>
<td>15, 12, 8%</td>
<td>33, 29, 23%</td>
</tr>
<tr>
<td>Fresh_1</td>
<td>12, 14, 10%</td>
<td>37, 43, 41%</td>
</tr>
<tr>
<td>Fresh_2</td>
<td>9, 8, 6%</td>
<td>31, 31, 29%</td>
</tr>
</tbody>
</table>

Next, we compare CT and online-MF in terms of their adaptation capabilities to new items. Figure 2 illustrates the cumulative density function (CDF) of the threads recommended by different methods against freshness. We can see that the CDF of CT increase sharply when freshness is large, which means that the probability of recommending fresh items is high. In other word, CT recommends much more fresh items than online-MF. Other than the quantity of recommending fresh items, the quality of such recommendations are crucial as well. Figure 3 shows the conditional success rate \( P(\text{Success}|\text{Freshness}) \) across different degrees of freshness for three courses. As the freshness increases, the conditional success rate of online-MF drops speedily while the CT method keeps a solid and stable performance across. It is obvious that CT outperforms online-MF by a large margin when freshness is high so that it is particularly strong for recommending fresh items. Fresh items are often not popular in terms of the total number of views at the time point of recommendation, however, there may be an ongoing trend within those fresh items. So identifying more fresh items accurately implies a strong adaptation power to new and evolving forum visiting patterns. This result validates our previous hypothesis that the CT recommender can adapt well to drifting user preferences.

4 Conclusion and Future Work

In this paper, we studied the adaptation of machine learning models from a sequential recommendation perspective. Through experimental analysis, both performance improvement and adaptation to drifting preferences are achieved using a new method called context tree. As a future work, exploratory algorithms are interesting to be tried. Furthermore, some partial context matching mechanisms can be investigated for allowing a mixture or generalization of similar paths and contexts.
References


