Hierarchical Reinforcement Learning for Course Recommendation in MOOCs

Jing Zhang, Bowen Hao, Bo Chen, Cuiping Li, Hong Chen
(Renmin University)
Jimeng Sun (Georgia Institute of Technology)
Course Recommendation

相关课程
计算机科学和Python编程导论（自主模式）
Python程序设计
MyCS：计算机科学入门
Problem Definition

Input: Historical enrolled courses of a user before $t$

- Machine Learning: Course at Stanford University
- Information Systems: Specialization at University of Minnesota
- Introduction to Classical Music: Course at Yale University

Output: Most possible courses to be enrolled at $t+1$

- Mathematics for Machine Learning: 3-course Specialization at Imperial College London
- Ancient Philosophy: Plato & His Predecessors at University of Pennsylvania
- Economics of Money and Banking at Columbia University
- Master of Public Health at University of Michigan
- Power Electronics: 6-course Specialization at University of Colorado Boulder
Challenges

• Users’ enrolled courses are usually diverse. The contributing courses may be diluted by noisy courses
• Even if no courses can contribute in predicting a random target course, each historical course will still be assigned an attention coefficient.
Data Analysis

<table>
<thead>
<tr>
<th>Users</th>
<th>Courses</th>
<th>Categories</th>
<th>User-course pairs</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>82,535</td>
<td>1,302</td>
<td>23</td>
<td>458,454</td>
<td>2016.10.1-2018.3.31</td>
</tr>
</tbody>
</table>

A large number of users enrolled diverse courses

(A bigger #categories/#courses indicates the user is more distractive)

The recommendation performance based on the diverse profiles is impacted
Idea to Deal with the Challenges

• Revise the user profiles by removing the noisy courses instead of assigning an attention coefficient to each of them.
  – But how to determine which courses should be removed?
  – Without the supervised information, can we automatically learn the pattern?
Solution

Revise the user profiles based on the feedbacks from the recommender

The revising process of a user profile: a sequential decision process

- The profile reviser (agent)
  - Revise the profile
  - Gets a delayed reward from the recommender
  - Update its policy

- The recommender (environment)
  - Update its parameters based on the profiles revised by the profile reviser
• A hierarchical Markov Decision Process
  • The agent firstly performs a high-level task to determine whether to revise the whole profile or not
  • If it decides to revise, the agent performs a low-level task of multiples actions to determine whether to remove each historical course or not
  • The overall task is finished after the low-level task is finished or the high-level task decides to make no revision.
The High-level Task

• Determine whether to revise the whole profile $\varepsilon^u$ or not
  
  – State:
    • The average cosine similarity between the embedding vectors of each historical course in $\varepsilon^u$ and the target course $c_i$.  
    • The average element wise product between the embedding vectors of each historical course in $\varepsilon^u$ and the target course $c_i$. 
    • The probability $P(y = 1|\varepsilon^u, c_i)$ of recommending $c_i$ to user $u$ by the basic recommender. 

    (The lower recommendation probability is, more effort should be taken to revise the profile)
  
  – Action:
    • {Revise, keep}
  
  – Delayed reward:

    Policy function: two-layer NN

    \[
    \begin{align*}
    H^l_t & = \text{ReLU}(W^l_1 s^l_t + b^l), \\
    \pi(s^l_t, a^l_t) & = P(a^l_t|s^l_t, \Theta^l) \\
    & = a^l_t \sigma(W^l_2 H^l_t) + (1 - a^l_t)(1 - \sigma(W^l_2 H^l_t)),
    \end{align*}
    \]

    The difference between the log-likelihood after and before the profile is revised.

    \[
    R(a^l_t, s^l_t) = \begin{cases} 
    \log p(\hat{E}^u, c_i) - \log p(\varepsilon^u, c_i), & \text{if } t = t_u; \\
    0, & \text{otherwise},
    \end{cases}
    \]
The low-level Task

- Determine whether to remove a historical course $e_t^u \in \varepsilon^u$ or not
  
  - **State:**
    - The cosine similarity between the embedding vectors of the current historical course $e_t^u$ and the target course $c_i$.
    - The element wise product between the embedding vectors of the current historical course $e_t^u$ and the target course $c_i$.
    - Effort taken in the course.
  
  - **Action:**
    - \{Remove, Keep\}
  
  - **Reward:**
    - Add an internal reward
      - speed up local learning and does not propagate to the high-level.
    - $G(a_t^l, s_t^l)$: calculate the average cosine similarity between each historical course and the target course after and before the profile is revised.

\[
R(a_t^m, s_t^m) + G(a_t^m, s_t^m)
\]

Effort: we calculate the ratio between the watch duration and the total duration of a video as the watch ratio, and use the maximal watch ratio of all the videos in a course to represent the effort taken by the user in the course.
Objective Function

- Maximize the expected reward:

\[ \Theta^* = \arg\max_\Theta \sum_\tau P_\Theta(\tau; \Theta) R(\tau), \]

A sequence of the sampled actions and the transited states:
\{s^l_1, a^l_1, s^l_2, \ldots, s^l_t, a^l_t, s^l_{t+1}, \ldots\}

- Update the policy network with policy gradient:

\[ \nabla_\Theta = \frac{1}{m} \sum_{m=1}^M \nabla_\Theta \log \pi_\Theta(s^m, a^m) R(a^m_t, s^m_t) \]
Training Procedure

Pre-train the basic recommendation model;
Pre-train the profiler reviser by running Algorithm 2
with the basic recommendation model fixed;
Jointly train the two models together by running
Algorithm 2;

Algorithm 1: The Overall Training Process

Input: Training data \( \{ \mathcal{E}^1, \mathcal{E}^2, \ldots, \mathcal{E}^{|U|} \} \), a pre-trained basic recommendation model and a profile reviser parameterized by \( \Theta^0 \) and \( \Phi^0 \) respectively

Initialize \( \Theta = \Theta^0, \Phi = \Phi^0 \);
for episode \( l = 1 \) to \( L \) do
  foreach \( \mathcal{E}^u := (e^u_1, \ldots, e^u_{|u|}) \) and \( c_i \) do
    Sample a high-level action \( a^h \) with \( \Theta^h \);
    if \( a^h = 0 \) then
      \( R(s^h, a^h) = 0 \)
    else
      Sample a sequence of low-level actions \( \{a^l_1, a^l_2, \ldots, a^l_{|u|} \} \) with \( \Theta^l \);
      Compute \( R(a^l_{i_u}, s^l_{i_u}) \) and \( G(a^l_{i_u}, s^l_{i_u}) \);
      Compute gradients by Eq. (5) and (6);
  end
end
Update \( \Theta \) by the gradients;
Update \( \Phi \) in the basic recommendation model;

Algorithm 2: The Hierarchical Reinforcement Learning

- Pre-train + joint train
- Sample actions and transited states and get rewards
- Update profile reviser and recommender
Experiment Results

Table 1: Recommendation performance (%).

<table>
<thead>
<tr>
<th>Methods</th>
<th>HR@5</th>
<th>HR@10</th>
<th>NDCG@5</th>
<th>NDCG@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPR</td>
<td>46.82</td>
<td>60.73</td>
<td>34.16</td>
<td>38.65</td>
</tr>
<tr>
<td>MLP</td>
<td>52.16</td>
<td>66.29</td>
<td>40.39</td>
<td>44.41</td>
</tr>
<tr>
<td>FM</td>
<td>46.01</td>
<td>61.07</td>
<td>35.28</td>
<td>40.15</td>
</tr>
<tr>
<td>FISM</td>
<td>52.73</td>
<td>65.64</td>
<td>40.00</td>
<td>44.98</td>
</tr>
<tr>
<td>GRU</td>
<td>52.07</td>
<td>68.63</td>
<td>38.92</td>
<td>46.30</td>
</tr>
<tr>
<td>NAIS</td>
<td>56.42</td>
<td>69.05</td>
<td>43.73</td>
<td>47.82</td>
</tr>
<tr>
<td>NASR</td>
<td>54.64</td>
<td>69.48</td>
<td>42.39</td>
<td>47.33</td>
</tr>
<tr>
<td>HRL+NAIS</td>
<td>64.59</td>
<td>79.68</td>
<td>45.74</td>
<td>50.69</td>
</tr>
<tr>
<td>HRL+NASR</td>
<td>59.05</td>
<td>74.50</td>
<td>47.51</td>
<td>52.73</td>
</tr>
</tbody>
</table>
Compared with One-level RL

- The average #Categories/#Courses of the revised profiles:
  - Two-level RL: 0.73
  - One-level RL: 0.75
- The revised profiles by the two-level HRL are more consistent

The high-level task of the two-level RL:
- The average #Categories/#Courses of
  - the kept profiles: 0.57
  - the revised profiles: 0.69
- High-level task tends to keep more consistent profiles

(A bigger #categories/#courses indicate the user is more distractive)
Compared with Greedy Revision

The greedy reviser
• firstly decides to revise the whole profile if $P(y = 1|\varepsilon^u, c_i) < \mu_1$
• and then removes the course $e_t^u \in \varepsilon^u$ if its cosine similarity with $c_i$ is less than $\mu_2$
Compared with Attentions

Table 2: Case studies of the profiles revised by HRL+NAIS and the attention coefficients learned by NAIS.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Revised profile or the learned attentions</th>
<th>The target course</th>
</tr>
</thead>
<tbody>
<tr>
<td>HRL+NAIS</td>
<td>Crisis Negotiation, Social Civilization, Web Technology, C++ Program</td>
<td>Web Development</td>
</tr>
<tr>
<td>NAIS</td>
<td>Crisis Negotiation(29.61), Social Civilization(29.09), Web Technology(28.32), C++ Program(28.12)</td>
<td>Web Development</td>
</tr>
<tr>
<td>HRL+NAIS</td>
<td>Modern Biology, Medical Mystery, Biomedical Imaging, R-Program</td>
<td>Biology</td>
</tr>
<tr>
<td>NAIS</td>
<td>Modern Biology(37.79), Medical Mystery(37.96), Biomedical Imaging(37.62), R Program(37.84)</td>
<td>Biology</td>
</tr>
<tr>
<td>HRL+NAIS</td>
<td>Web Technology, Art Classics, National Unity Theory, Philosophy</td>
<td>Life Aesthetics</td>
</tr>
<tr>
<td>NAIS</td>
<td>Web Technology(38.32), Art Classics(35.87), National Unity Theory(40.63), Philosophy(43.69)</td>
<td>Life Aesthetics</td>
</tr>
</tbody>
</table>
Conclusion

• We present the first attempt to solve the problem of course recommendation in MOOCs platform by a hierarchical RL model.

• The model jointly trains a profile reviser and a basic recommendation model, which enables the hierarchical RL model effectively to remove the noisy courses to the target course, and enables recommendation model to be improved on revised user profiles by an agent.

• The experimental results on a real dataset collected from XuetangX validate the effectiveness of the proposed model.
Thank you!