Explainable Clustering and Cluster-based Collaborative Filtering

Yuntao Du and Jianxun Lian
Microsoft Research Asia
2023/1/10
Outline

- Explainable Clustering
  - Why clustering?
  - Why explainability is matter for clustering algorithms?
  - How to achieve explainable clustering?

- Cluster-based Collaborative Filtering
  - Revisiting: collaborative filtering as an explicit/implicit clustering process
  - Naïve clustering approach for collaborative filtering
  - Our work: Towards Explainable Collaborative Filtering with Taste Clusters Learning
Part 1: Explainable Clustering
Clustering

Why clustering?

- One of the biggest topics in data science
- Clustering is to identify patterns or discover structural properties in a data set by quantizing the unlabeled points
  - Discover coherent groups among a supermarket’s customers
  - Find friends with similar habits / tastes

A classical problem, but still plenty need to be done…

- Classical clustering algorithms (k-means, DBSCAN, etc)
- Co-clustering / constrained clustering / multi-view clustering
- Deep clustering…
Explainable Clustering

- Explainability for clustering
  - Cluster process may be determined using all the features of the data or embeddings
    - Why they need to be in the same cluster?
  - Cluster results can be hard to explain
    - What’s the meaning of the cluster?
  - Quality is not the only objective in many fields like healthcare

- Goal
  - Provide provable insight into what parts of the data the cluster algorithm used to make its prediction
  - Achieve good balance between quality and explainability
Tree-based explainable clustering

- **k-means (reference clustering)**
  - Cluster goal: minimize *k*-means cost (NP-hard)

  \[
  \text{cost}(C) = \sum_{i=1}^{k} \sum_{x \in C^i} ||x - \text{mean}(C^i)||^2
  \]

- **Naive explanation**
  - Directly using cluster centers as explanation
  - Depends on all data points and all the features in a complicated way

Tree-based explainable clustering

**Goal**
- Explainable by design (self-explainable)
- Only look at some determined features to make clusters
  - Not depend on the cluster centers
- At each step, split only one feature with threshold
  - Leaves correspond to clusters (same as decision tree)

Tree-based explainable clustering

- General scheme
  - Find a clustering using some clustering algorithm
  - **Label** each example according to its cluster
  - Call a **supervised** algorithm that learns a decision tree

- ID3/C4.5 algorithm?
  - Split according to the information gain, no good

\[
Gain(D, a) = Ent(D) - \sum_{t=1}^{T} \frac{|D^t|}{|D|} Ent(D^t)
\]

Tree-based explainable clustering

- **Iterative Mistake Minimization (IMM)**
  - Mistake
    - A point $x$ is a mistake for node $u$ if $x$ and its center $c(x)$ reached and then separated by $u$’s split
  - Each step we take the split (i.e., feature and threshold) that minimizes mistake

Tree-based explainable clustering

- **Iterative Mistake Minimization (IMM)**
  - As long as there is more than one center, find the split with minimal number of mistakes

Tree-based explainable clustering

(IMM Properties)

- Running time
  - $O(kdn\log(n))$
  - For each of the $k - 1$ inner nodes and each of the $d$ features, we can find the split that minimizes the number of mistakes for this node and feature, in time $O(n\log(n))$
  - Comparable to standard k-means $O(tkdn)$

- Approximation factor

<table>
<thead>
<tr>
<th></th>
<th>$k$-medians</th>
<th>$k$-means</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k = 2$</td>
<td>$k &gt; 2$</td>
<td>$k = 2$</td>
</tr>
<tr>
<td>$k &gt; 2$</td>
<td>$\Omega(\log k)$</td>
<td>$3(1 - \frac{1}{d})^2$</td>
</tr>
<tr>
<td>Lower</td>
<td>$2 - \frac{1}{d}$</td>
<td>$\Omega(\log k)$</td>
</tr>
<tr>
<td>Upper</td>
<td>$2$</td>
<td>$O(k)$</td>
</tr>
</tbody>
</table>

https://ucsdml.github.io/jekyll/update/2020/10/16/explain_2_means.html
Tree-based explainable clustering

- Improvements 1
  - What if we can have $k'$ leaves ($k' > k$)?
  - Flexible trade-off between explainability & accuracy

Tree-based explainable clustering

- **Improvements 2**
  - Can we make full use of the reference clustering?
    - Select cuts based on centers, not data points
    - Only need to scan data once, nearly zero computational overhead

---

**Algorithm 1:** Explainable $k$-medians algorithm.

1. **Input:** A collection of $k$ centers $\mathcal{U} = \{\mu^1, \mu^2, \ldots, \mu^k\} \subset \mathbb{R}^d$.
2. **Output:** A threshold tree with $k$ leaves.
3. Leaves $\leftarrow \{\mathcal{U}\}$
4. while $|\text{Leaves}| < k$ do
   5. Sample $(i, \theta)$ uniformly at random from AllCuts.
   6. for each $B \in \text{Leaves}$ that are split by $(i, \theta)$ do
      7. Split $B$ into $B^-$ and $B^+$ and add them as left and right children of $B$.
      8. Update Leaves.
6. return the threshold tree defined by all cuts that separated some $B$.

Tree-based explainable clustering

- **Improvements 2**
  - The constructed decision tree is almost good enough

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>$k$-medians</th>
<th>$k$-means</th>
<th>$\ell_p$-norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(k)$</td>
<td>$O(k^2)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$O(d \log k)$</td>
<td>$O(kd \log k)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$O(\log^2 k)$</td>
<td>$O(k \log^2 k)$</td>
<td>$O(k^{p-1} \log^2 k)$</td>
<td></td>
</tr>
<tr>
<td>$O(\log k \log \log k)$</td>
<td>$O(k \log k \log \log k)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$O(\log k \log k)$</td>
<td>$O(k \log k)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$O(d \log^2 d)$</td>
<td>$O(k^{1-2/p} \text{polylog } k)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lower bounds</th>
<th>$\Omega(\log k)$</th>
<th>$\Omega(\log k)$</th>
<th>$\Omega(k^{p-1})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Omega(\log k)$</td>
<td>$\Omega(\log k)$</td>
<td>$\Omega(k)$</td>
<td>$\Omega(k^{p-1})$</td>
</tr>
<tr>
<td>$\Omega(k/\log k)$</td>
<td>$\Omega(k/\log k)$</td>
<td></td>
<td>$\Omega(k)$</td>
</tr>
<tr>
<td>$\Omega(\min(d, \log k))$</td>
<td>$\Omega(k)$</td>
<td>$\Omega(\min(d, \log k))$</td>
<td>$\Omega(k)$</td>
</tr>
<tr>
<td>$\Omega(k^{1-2/p} / \text{polylog } k)$</td>
<td></td>
<td>$\Omega(k^{1-2/p} / \text{polylog } k)$</td>
<td></td>
</tr>
</tbody>
</table>

**This paper**

Dasgupta et al. [6]

Laber and Murtinho [10]

Makarychev and Shan [12]

Esfandiari et al. [7]

Charikar and Hu [5]

Tree-based explainable clustering

- **Improvements 3**
  - Can perform clustering and decision tree training **holistically**?
    - Optimize the decision tree’s size (for explainability) and the distortion (for accuracy) together
    - Assume two groups of features: accuracy features and explainability features
    - Clustering with accuracy features, analyze with explainability features

Problem setting

➢ Define similarity functions as the combination of two measures

\[
\frac{(1 - \alpha) \times \text{a-distance}}{\max\{\text{All a-distances}\}} + \frac{\alpha \times \text{e-distance}}{\max\{\text{All e-distances}\}}
\]

➢ Can use any distance function (DTW) and clustering method (k-medoids)
➢ Agnostic to the decision tree training algorithm

Goal

\[
\min_{k, \alpha} D(k, \alpha) + \lambda N(k, \alpha)
\]

➢ \(D\): cluster distortion where a lower value is better as we would like the clusters to be coherent
➢ \(N\): number of decision tree nodes

Tree-based explainable clustering

- Monotonicity properties
  - As $k$ increases, $D$ is decreasing while $N$ is increasing
  - As $\alpha$ increases, $D$ is increasing while $N$ is decreasing
  - Given $[k_1, k_2]$ and $[\alpha_1, \alpha_2]$, we have
    \[
    D(k_2, \alpha) + \lambda N(k, \alpha) \\
    \geq D(k_2, \alpha) + \lambda N(k_1, \alpha) \\
    \geq D(k_2, \alpha_1) + \lambda N(k_1, \alpha_2).
    \]
  - Get the upper bound and lower bound for $D(k, \alpha) + \lambda N(k, \alpha)$
  - Search for the desired parameters

Tree-based explainable clustering

- Search for the $k$ and $\alpha$ parameters

---

**Algorithm 1: XClusters algorithm**

**Input:** training data $S$, maximum $k$ value $k_{\text{max}}$

**Parameters:** $k$, $\alpha$

**Output:** clusters and decision tree

1. $B \leftarrow [(1, 0), (k_{\text{max}}, 1)]$
2. Compute upper and lower bounds of $B$
3. $B^* \leftarrow B$
4. $Q.push(B)$
5. while $\neg Q.empty()$ do
6. $B \leftarrow Q.pop()$ // Block with lowest lower bound
7. if $B$'s normalized $k$ width is longer than the normalized $\alpha$ width then
8. \{ $B_1, B_2$ \} $\leftarrow$ Split $B$ by $k$ into two blocks
9. else
10. \{ $B_1, B_2$ \} $\leftarrow$ Split $B$ by $\alpha$ into two blocks
11. Compute upper and lower bounds of $B_1$ and $B_2$
12. $Q.push(\{B_1, B_2\})$
13. if $\min_{B \in Q} B.upper() < B^*.upper()$ then
14. $B^* \leftarrow \arg \min_{B \in Q} B.upper()$
15. $Q \leftarrow Q \setminus \{B' \in Q | B'.lower() + \epsilon_b \geq B^*.upper()\}$
16. return Clusters and decision tree of $B^*.upper()$

---

Figure 2: The XClusters algorithm iteratively splits blocks while pruning blocks that are not worth exploring based on their lower and upper bounds.
Tree-based explainable clustering

- Experiments
  - Use three time series dataset: Credit, COVID-19, contracts
  - Accuracy features: time-series trends features
  - Explainability features: demographics information
  - Achieve good balance between accuracy and explainability, as well as efficiency

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>$D + \lambda N$</th>
<th>$D$</th>
<th>$N$</th>
<th>Runtime (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Credit Card</td>
<td>2-Step</td>
<td>0.981±0.000</td>
<td>0.206±0.000</td>
<td>0.775±0.000</td>
<td>0.011±0.001</td>
</tr>
<tr>
<td></td>
<td>GS</td>
<td>0.808±0.000</td>
<td>0.282±0.000</td>
<td>0.526±0.000</td>
<td>1.413±0.023</td>
</tr>
<tr>
<td></td>
<td>BO</td>
<td>0.851±0.028</td>
<td>0.238±0.030</td>
<td>0.613±0.045</td>
<td>1.416±0.158</td>
</tr>
<tr>
<td></td>
<td>XClusters</td>
<td>0.815±0.000</td>
<td>0.270±0.000</td>
<td>0.544±0.000</td>
<td>0.321±0.004</td>
</tr>
<tr>
<td>DS4C</td>
<td>2-Step</td>
<td>0.702±0.000</td>
<td>0.103±0.000</td>
<td>0.599±0.000</td>
<td>0.008±0.001</td>
</tr>
<tr>
<td></td>
<td>GS</td>
<td>0.406±0.000</td>
<td>0.127±0.000</td>
<td>0.279±0.000</td>
<td>1.345±0.012</td>
</tr>
<tr>
<td></td>
<td>BO</td>
<td>0.415±0.026</td>
<td>0.126±0.005</td>
<td>0.289±0.031</td>
<td>1.546±0.225</td>
</tr>
<tr>
<td></td>
<td>XClusters</td>
<td>0.466±0.000</td>
<td>0.128±0.000</td>
<td>0.338±0.000</td>
<td>0.308±0.004</td>
</tr>
<tr>
<td>Contracts</td>
<td>2-Step</td>
<td>1.018±0.000</td>
<td>0.216±0.000</td>
<td>0.802±0.000</td>
<td>0.011±0.000</td>
</tr>
<tr>
<td></td>
<td>GS</td>
<td>0.778±0.000</td>
<td>0.228±0.000</td>
<td>0.550±0.000</td>
<td>1.974±0.000</td>
</tr>
<tr>
<td></td>
<td>BO</td>
<td>0.827±0.032</td>
<td>0.301±0.057</td>
<td>0.526±0.046</td>
<td>1.550±0.240</td>
</tr>
<tr>
<td></td>
<td>XClusters</td>
<td>0.619±0.000</td>
<td>0.442±0.000</td>
<td>0.177±0.000</td>
<td>0.349±0.004</td>
</tr>
</tbody>
</table>

Tree-based explainable clustering

- Case study

Explainable clustering

Future direction

- Efficiency
  - Can we explore parallelizations for threshold tree construction?

- Generalization
  - Can we allow each node to be a hyperplane in a chosen number of dimensions instead of only splitting along one feature?

- Evaluation
  - How to evaluate the quality of explainability?

- Other explainable approaches
  - Can we go beyond the decision tree style for clustering explanation?
Part2: Cluster-based Collaborative Filtering
CF as a clustering process

- **Rationale behind collaborative filtering**
  - Finding like-minded users for a (group of) target user(s) to share preferences
  - Much like clustering!

- **Both memory-based CF (kNN, itemCF) and model-based CF (MF, NCF) perform clustering explicitly or implicitly**
  - Directly cluster similar users/items into groups, then perform kNN for recommendation
  - Learn to group similar users/items implicitly, then select most similar items for recommendation
CF as a clustering process

- Naïve approach
  - Clustering users with any cluster algorithm
  - Data smoothing with cluster-specific rating (optional)
    \[
    \hat{R}_u(t) = \overline{R_u} + \Delta R_{C_u}(t) \quad \Delta R_{C_u}(t) = \sum_{u \in C_u(t)} (R_{u'}(t) - \overline{R_u}) / |C_u(t)|
    \]
  - Neighbor pre-selection with clusters
    - Select most similar group for active user
  - Neighbor selection
    - Select most similar users from certain group
  - Prediction
    \[
    R_{u_a}(t) = \overline{R_{u_a}} + \sum_{i=1}^{K} w_{u_t} \cdot \text{sim}_{u_a,u} \cdot (R_{u}(t) - \overline{R_u}) \quad \text{sim}_{u_a,C} = \frac{\sum_{i \in T(u_a) \cap T(C)} \Delta R_C(t) \cdot (R_{u_a}(t) - \overline{R_u})}{\sqrt{\sum_{i \in T(u_a) \cap T(C)} (\Delta R_C(t))^2} \cdot \sqrt{\sum_{i \in T(u_a) \cap T(C)} (R_{u_a}(t) - \overline{R_u})^2}}.
    \]

- Is it necessary to explicitly perform clusters NOW?
  - Yes!

CF as a clustering process

- A toy movie recommendation scenario

<table>
<thead>
<tr>
<th></th>
<th>Andre</th>
<th>Star Wars</th>
<th>Batman</th>
<th>Rambo</th>
<th>Hiver</th>
<th>Whispers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lyle</td>
<td>y</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ellen</td>
<td>y</td>
<td>y</td>
<td></td>
<td></td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>Fred</td>
<td>y</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dean</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jason</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>y</td>
<td>y</td>
</tr>
</tbody>
</table>
**CF as a clustering process**

- **Rearrange the table…**

<table>
<thead>
<tr>
<th></th>
<th>Batman</th>
<th>Rambo</th>
<th>Andre</th>
<th>Hiver</th>
<th>Whispers</th>
<th>Star Wars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lyle</td>
<td>y</td>
<td></td>
<td></td>
<td>y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ellen</td>
<td>y</td>
<td>y</td>
<td></td>
<td>y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jason</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fred</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>y</td>
</tr>
<tr>
<td>Dean</td>
<td>y</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td>y</td>
</tr>
</tbody>
</table>
CF as a clustering process

- Rearrange the table…

<table>
<thead>
<tr>
<th>intellectual</th>
<th>action</th>
<th>foreign</th>
<th>classical</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Batman</td>
<td>Rambo</td>
<td>Andre</td>
</tr>
<tr>
<td>Lyle</td>
<td></td>
<td></td>
<td>y</td>
</tr>
<tr>
<td>Ellen</td>
<td></td>
<td></td>
<td>y</td>
</tr>
<tr>
<td>Jason</td>
<td></td>
<td></td>
<td>y</td>
</tr>
<tr>
<td>fun</td>
<td>Fred</td>
<td></td>
<td>y</td>
</tr>
<tr>
<td></td>
<td>Dean</td>
<td></td>
<td>y</td>
</tr>
</tbody>
</table>
CF as a clustering process

- Rearrange the table...

<table>
<thead>
<tr>
<th></th>
<th>action</th>
<th>foreign</th>
<th>classical</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Batman</td>
<td>Rambo</td>
<td>Andre</td>
</tr>
<tr>
<td>Lyle</td>
<td></td>
<td></td>
<td>y</td>
</tr>
<tr>
<td>Ellen</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jason</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fred</td>
<td>y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dean</td>
<td>y</td>
<td>y</td>
<td></td>
</tr>
</tbody>
</table>

- How to achieve this?
Towards Explainable Collaborative Filtering with Taste Clusters Learning

Yuntao Du\textsuperscript{1}, Jianxun Lian\textsuperscript{2}, Jing Yao\textsuperscript{2}, Xiting Wang\textsuperscript{2}, Mingqi Wu\textsuperscript{3}, Lu Chen\textsuperscript{1}, Yunjun Gao\textsuperscript{1} and Xing Xie\textsuperscript{2}

\textsuperscript{1}College of Computer Science, Zhejiang University, China
\textsuperscript{2}Microsoft Research Asia, China
\textsuperscript{3}Microsoft Gaming, USA

The WebCof 2023, Under Review
Motivations

- Explainable Recommendation
  - A growing need to ensure that the users understand and trust the system
  - Explanations serve as a bridge between recommender systems and users/developers
    - Increase user trust
    - Help users make better decisions (satisfactions)
    - Persuade users to try or buy an item (persuasiveness)
    - Assisting developers in model debugging and abnormal case studies

Explanations: why the items are recommended
Desirable Properties for Explainable CF

- **Flexibility**
  - The dimension of latent embeddings and the number of interpretable features/topics do not necessarily match ([1] fails on this)

- **Coherence**
  - A model’s interpretable modules and predictive modules should be aligned during predictive decision making rather than being decoupled as independent modules ([2] fails on this)

- **Self-explainable**
  - A model can provide interpretable clues that truly reveal the model’s running logic, instead of learning a post-hoc model for explanation ([3] fails on this)

Our Method

- Explainable Collaborative Filtering (ECF)
  - The first framework that satisfies all three properties
  - Core idea: mining various taste clusters, and map users/items to corresponding clusters
  - Taste clusters: A group of items which are not only similar in users’ latent interest space, but also explicitly share some common tags
Recommendation process of ECF

- **Item recommendation**
  - Prediction score of user $u$ and item $i$ can be calculated by multiplying their affiliations with taste clusters
    \[
    \hat{y}_{ui} = \text{sparse\_dot}(a_u, x_i),
    \]

- **Personalized explanation**
  - For each prediction $\hat{y}_{ui}$, ECF is able to generate explanation by measuring the coherence between users’ and items’ taste cluster affiliations:
    \[
    C_{ui} = S(a_u) \cap S(x_i),
    \]
  - And importance score $w^c_{ui}$ is introduced to quantify the contribution of each taste cluster in $C_{ui}$:
    \[
    w^c_{ui} = a_{uc} \times x_{ic}.
    \]
Learning Sparse Affiliation

- Directly learning the affiliation matrix from data is hard
  - Due to its sparsity nature for readability

- Initialize the users/items and taste clusters with embedding

\[ \tilde{x}_{ic} = \cos(v_i, h_c), \]

\[ m_{ic} = \begin{cases} 
1 & \text{if } c \in \text{argTopm}(\tilde{x}_i) \\
0 & \text{otherwise} 
\end{cases} \]

\[ x_i = \sigma(\tilde{x}_i) \odot m_i, \]

- Learn it with reparameterized trick

\[ m_{ic} \approx \hat{m}_{ic} = \frac{\exp(\cos(v_i, h_c)/T)}{\sum_c \exp(\cos(v_i, h_c)/T)}, \]

\[ \hat{m}_{ic} = \hat{m}_{ic} + \text{detach_gradient}(m_{ic} - \hat{m}_{ic}), \]
Optimization of ECF

- **Reconstruction Loss**
  - Using user/item-cluster affiliations for prediction:
    \[ L_{CS} = \sum_{(u,i,j) \in O} -\ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}), \quad \hat{y}_{ui} = \text{sparse\_dot}(a_u, x_i), \]

- **Tag Similarity Loss**
  - The items in the same taste clusters should share the similar tags
  - Using TF-IDF style to select informative tags for taste clusters:
    \[ d_{ct} = \tilde{d}_{ct} \times \log\left(\frac{N}{f_t + \epsilon}\right), \quad \beta_{ct} = \frac{\exp(d_{ct}/\tau)}{\sum_{c_j \in T} \exp(d_{ct}/\tau)}, \]
    - Maximizing the likelihood of the probabilities of Top-\( P \) tags so that the taste clusters can be easily interpreted by those tags:
      \[ L_{TS} = \sum_{c \in C} \sum_{t \in \text{argTopP}(\beta_c)} -\log \beta_{ct}, \]

- **Independence Loss**
  - Taste clusters should be different to present different user interest space:
    \[ L_{IND} = \sum_{c \in C} -\log \frac{\exp(s(h_c, h_c))}{\sum_{c' \in C} \exp(s(h_c, h_{c'}))}, \]
Optimization of ECF

- **Learning taste cluster from three aspects**
  - No need to tune the weight for each loss
    \[ \mathcal{L}_{TC} = \mathcal{L}_{CS} + \mathcal{L}_{TS} + \mathcal{L}_{IND}. \]

- **ECF loss**
  - Directly learning the taste cluster is hard to converge since the supervised signals are sparse
  - Adding auxiliary supervised signals from user-item predictions
    \[ \mathcal{L}_{CF} = \sum_{(u,i,j) \in O} -\ln \sigma(e_u^T v_i - e_u^T v_j), \]
  - Embeddings can be learned from any embedding-based models (MF for simplicity)
  - Learn ECF with guidance from auxiliary collaborative signals
    \[ \mathcal{L}_{ECF} = \mathcal{L}_{TC} + \lambda \mathcal{L}_{CF}, \]
Forest Mechanism

Observation

- Sparse affiliations between user/item and clusters would inevitably harm recommendation accuracy
- We do not know how many clusters needed to model users’ hidden interest space properly

Forest mechanism for ECF

- We randomly select $|C|$ items and use different random seeds for model training
- Train $F$ different instances to form the final ECF model, and the final prediction is based on the summation of all $M$ models
- Boost the performance and provide a comprehensive explanation for predictions
Metrics for Explainability

- **In-cluster item coverage**
  - The proportion of items in the taste cluster that the selected tags can cover
  \[
  \text{Cov.} = \frac{1}{Z} \sum_{c \in C} \sum_{i \in c} \frac{1 (T_i \cap T_c)}{|c|},
  \]

- **Tag utilization**
  - How many unique tags are used for interpreting taste clusters
  \[
  \text{Util.} = \frac{1}{|\mathcal{T}|} \bigcup_{c \in C} \mathcal{T}_c,
  \]

- **Silhouette**
  - Similarity difference between intra-cluster items and inter-cluster items
  \[
  \text{Sil.} = \frac{1}{|I|} \sum_{i \in I} \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}.
  \]

- **Informativeness**
  - Distinctiveness of selected tags to represent the items in the taste cluster
  \[
  \text{Info.} = \frac{1}{|C|} \sum_{c_i \in C} \frac{|R(T_c) \cap c_i|}{|c|},
  \]
Metrics for Explainability (Cont.)

- Human evaluation
  - 30 volunteers evaluating the explainability of both taste clusters (Task 1) and user-item recommendations (Task 2)
  - Each volunteer is asked to look items’ profiles and user’s interactions
  - Then evaluate the results by comparison with baselines
  - Task 1
    - Rank the quality of generated clusters' tags
  - Task 2
    - Rank the quality of user-item explanation
Experimental Evaluation

- **Datasets**
  - Real-world datasets (Xbox) and public datasets (MovieLens and Last-FM)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Users</th>
<th>#Items</th>
<th>#Interactions</th>
<th>#Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xbox</td>
<td>465,258</td>
<td>330</td>
<td>6,240,251</td>
<td>115</td>
</tr>
<tr>
<td>MovieLens</td>
<td>6,033</td>
<td>3,378</td>
<td>836,434</td>
<td>18</td>
</tr>
<tr>
<td>Last-FM</td>
<td>53,486</td>
<td>2,062</td>
<td>2,228,949</td>
<td>54</td>
</tr>
</tbody>
</table>

- **Recommendation performance**
  - Achieve excellent accuracy performance while providing interpretability
  - Our method greatly outperforms the baseline in all metrics across all datasets

<table>
<thead>
<tr>
<th></th>
<th>R@5</th>
<th>R@10</th>
<th>N@5</th>
<th>N@10</th>
<th>R@5</th>
<th>R@10</th>
<th>N@5</th>
<th>N@10</th>
<th>R@5</th>
<th>R@10</th>
<th>N@5</th>
<th>N@10</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Xbox</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MF</td>
<td>0.2615</td>
<td>0.3686</td>
<td>0.2383</td>
<td>0.2824</td>
<td>0.0601</td>
<td>0.0975</td>
<td>0.2738</td>
<td>0.2511</td>
<td>0.0289</td>
<td>0.0446</td>
<td>0.0428</td>
<td>0.0443</td>
</tr>
<tr>
<td>NCF</td>
<td>0.2372</td>
<td>0.3433</td>
<td>0.2065</td>
<td>0.2503</td>
<td>0.0594</td>
<td>0.0985</td>
<td>0.2701</td>
<td>0.2517</td>
<td>0.0269</td>
<td>0.0456</td>
<td>0.0396</td>
<td>0.0383</td>
</tr>
<tr>
<td>CDAE</td>
<td>0.2604</td>
<td>0.3738</td>
<td>0.2346</td>
<td>0.2813</td>
<td>0.0609</td>
<td>0.0946</td>
<td>0.2671</td>
<td>0.2534</td>
<td>0.0286</td>
<td>0.0402</td>
<td>0.0431</td>
<td>0.0518</td>
</tr>
<tr>
<td>LightGCN</td>
<td>0.2684</td>
<td>0.3625</td>
<td>0.2382</td>
<td>0.2837</td>
<td>0.0699</td>
<td>0.1163</td>
<td>0.2979</td>
<td>0.2752</td>
<td>0.0398</td>
<td>0.0578</td>
<td>0.0605</td>
<td>0.0634</td>
</tr>
<tr>
<td><strong>MovieLen</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MF</td>
<td>0.2647</td>
<td>0.3652</td>
<td>0.2368</td>
<td>0.2873</td>
<td>0.0657</td>
<td>0.1027</td>
<td>0.2866</td>
<td>0.2635</td>
<td>0.0319</td>
<td>0.0482</td>
<td>0.0471</td>
<td>0.0484</td>
</tr>
<tr>
<td>NCF</td>
<td>0.2601</td>
<td>0.3613</td>
<td>0.2355</td>
<td>0.2806</td>
<td>0.0603</td>
<td>0.0986</td>
<td>0.2719</td>
<td>0.2498</td>
<td>0.0295</td>
<td>0.0488</td>
<td>0.0456</td>
<td>0.0457</td>
</tr>
<tr>
<td><strong>Last-FM</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MF</td>
<td>0.2907</td>
<td>0.3983</td>
<td>0.2615</td>
<td>0.3159</td>
<td>0.0787</td>
<td>0.1276</td>
<td>0.3122</td>
<td>0.2911</td>
<td>0.0374</td>
<td>0.0548</td>
<td>0.0562</td>
<td>0.0594</td>
</tr>
<tr>
<td>ECF single</td>
<td>0.1714</td>
<td>0.2763</td>
<td>0.1423</td>
<td>0.1854</td>
<td>0.0352</td>
<td>0.0608</td>
<td>0.1584</td>
<td>0.1505</td>
<td>0.0205</td>
<td>0.0315</td>
<td>0.0339</td>
<td>0.0345</td>
</tr>
<tr>
<td>ECF</td>
<td><strong>0.2970</strong></td>
<td><strong>0.4299</strong></td>
<td><strong>0.2644</strong></td>
<td><strong>0.3193</strong></td>
<td>0.0788</td>
<td>0.1325</td>
<td>0.3183</td>
<td>0.2952</td>
<td><strong>0.0455</strong></td>
<td><strong>0.0635</strong></td>
<td><strong>0.0782</strong></td>
<td><strong>0.0749</strong></td>
</tr>
</tbody>
</table>
Experimental Evaluation

- **Explainability**
  - **K-means**: similarity-oriented method which utilizes item embedding from MF to perform K-means algorithm
  - **TagCluster**: tag-oriented method which collects items with the same tags
  - **ECF** takes all aspects into consideration so that it can avoid obvious shortcomings on a certain metric

<table>
<thead>
<tr>
<th>Method</th>
<th>Cov.</th>
<th>Util.</th>
<th>Sil.</th>
<th>Info.</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Xbox</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECF</td>
<td>0.8002</td>
<td>0.7052</td>
<td>0.2604</td>
<td>0.3162</td>
<td>1.7463</td>
</tr>
<tr>
<td>TagCluster</td>
<td>0.9950</td>
<td>0.2878</td>
<td>-0.1788</td>
<td>0.1579</td>
<td>0.9262</td>
</tr>
<tr>
<td>K-means</td>
<td>0.5710</td>
<td>0.3739</td>
<td>0.4286</td>
<td>0.0185</td>
<td>1.0563</td>
</tr>
<tr>
<td>Random</td>
<td>0.5396</td>
<td>0.1450</td>
<td>-0.3614</td>
<td>0.0125</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>RankOfTask1</th>
<th>RankOfTask2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECF</td>
<td>1.73</td>
<td>1.3</td>
</tr>
<tr>
<td>TagCluster</td>
<td>2</td>
<td>2.5</td>
</tr>
<tr>
<td>K-means</td>
<td>2.73</td>
<td>2.23</td>
</tr>
<tr>
<td>Random</td>
<td>3.63</td>
<td>3.93</td>
</tr>
</tbody>
</table>
Experimental Evaluation

- Ablation Study
  - Impact of top-\(m\) and top-\(n\) selection

![Graphs showing recall@20 for Last-FM and Xbox datasets with varying number of affiliations.](image)
Experimental Evaluation

- Ablation Study
  - Impact of the auxiliary collaborative signals $\lambda$

![Graphs showing performance metrics vs. $\lambda$ for different datasets](image-url)
Experimental Evaluation

- Ablation Study
  - Impact of the forest mechanism

![Graphs showing performance metrics](a) Xbox dataset  
(b) MovieLens dataset
Case Study: Last-FM

- **Learned Taste Clusters**
  - Can be used to correct tags
    - Tags for *Bubbly*: `female_vocalists|pop|folk|acoustic|love`
    - Missing tag `singer_songwriter`
      - Colbie Caillat is also a songwriter who wrote the song
Case Study: Last-FM

- Explanations of the recommendation
  - The weights of affiliation matrix indicate the relatedness between users/items with taste clusters
  - Find the explanation paths for prediction score
    - \( i_{71} \rightarrow c_1 \rightarrow i_{77414} \) and \( i_{71} \rightarrow c_4 \rightarrow i_{77414} \)
Case Study: Xbox Game Pass

- Real-world dataset with small items/games (~400)

- Recommendation accuracy

<table>
<thead>
<tr>
<th>model</th>
<th>Recall@5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>0.02</td>
</tr>
<tr>
<td>MF</td>
<td>0.18</td>
</tr>
<tr>
<td>ECF&lt;sub&gt;single&lt;/sub&gt;</td>
<td>0.14</td>
</tr>
<tr>
<td>ECF</td>
<td>0.24</td>
</tr>
</tbody>
</table>
Case Study: Xbox Game Pass

- Real-world dataset with small items/games (~400)

- Learned Taste Clusters

<table>
<thead>
<tr>
<th>ClusterID</th>
<th>Tags</th>
<th>Hard Count</th>
<th>HalfHard Count</th>
<th>Count</th>
<th>In_sim</th>
<th>Cross_sim</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>Difficulty_Low</td>
<td>LearningCurve_Low</td>
<td>GoodForKids</td>
<td>MoodsMotivations_Cozy</td>
<td>6</td>
<td>10</td>
</tr>
</tbody>
</table>
## Case Study: Xbox Game Pass

- **Affiliated games**

<table>
<thead>
<tr>
<th>ClusterID, TitleID, Game, Hard</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>36, 1970834532, RAGE 2 (PC), True</td>
<td>36, 1891258006, The Bard's Tale ARPG : Remastered and Resnarkled, False</td>
<td>36, 2102421406, Day of the Tentacle Remastered, False</td>
<td>36, 2096242259, ANVIL : Vault Breaker (Game Preview), False</td>
</tr>
<tr>
<td>36, 2055724194, SHENZHEN I/O, False</td>
<td>36, 1696012554, Goat Simulator Windows 10, False</td>
<td>36, 1621285366, DOOM (1993), True</td>
<td>36, 1843641391, Quake, True</td>
</tr>
<tr>
<td>36, 2090892978, Farming Simulator 22, False</td>
<td>36, 1805483741, Dishonored®: Death of the Outsider™ (PC), False</td>
<td>36, 2078926688, DOOM II (Classic), True</td>
<td>36, 167705209, Wolfenstein: The New Order (PC), False</td>
</tr>
<tr>
<td>36, 682562723, Halo: Spartan Assault, False</td>
<td>36, 1822205071, The Anacrusis (Game Preview), False</td>
<td>36, 1909396590, Wolfenstein: The Old Blood (PC), False</td>
<td></td>
</tr>
</tbody>
</table>
More Applications of ECF

- Taste Cluster Recommendation
  - A new but ubiquitous recommendation task like playlist recommendation in Spotify or gamelist recommendation in Xbox

- User Profiling
  - User-cluster affiliations discovered by ECF can also be used as user profiles directly
  - Can be used for user-level predictive tasks, ad audience targeting and look-alike audience extension, etc.

- Flexibility
  - Applied with other popular embedding-based methods like LightGCN
Future direction

➢ Optimization
  - How to optimize taste clusters in an elegant way?

➢ Quality
  - How to tag taste clusters properly and improve the quality?

➢ Scalability
  - How to apply ECF to scenarios with millions of users and items?

➢ Generalization
  - Can we go beyond item tags? Knowledge graph, reviews…
Thank you!

Questions?
Yuntao Du
ytdu@zju.edu.cn