Decision Tree and Instance-Based Learning for Label Ranking

Weiwei Cheng, Jens Hühn and Eyke Hüllermeier
Knowledge Engineering & Bioinformatics Lab
Department of Mathematics and Computer Science
University of Marburg, Germany
Label Ranking (an example)

Learning customers’ preferences on cars:

<table>
<thead>
<tr>
<th>customer 1</th>
<th>label ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>MINI &gt; Toyota &gt; BMW</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>customer 2</th>
<th>label ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMW &gt; MINI &gt; Toyota</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>customer 3</th>
<th>label ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMW &gt; Toyota &gt; MINI</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>customer 4</th>
<th>label ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toyota &gt; MINI &gt; BMW</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>new customer</th>
<th>label ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>???</td>
<td></td>
</tr>
</tbody>
</table>

where the customers could be described by feature vectors, e.g., (gender, age, place of birth, has child, ...)

1/17
**Label Ranking (an example)**

Learning customers’ preferences on cars:

<table>
<thead>
<tr>
<th></th>
<th>MINI</th>
<th>Toyota</th>
<th>BMW</th>
</tr>
</thead>
<tbody>
<tr>
<td>customer 1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>customer 2</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>customer 3</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>customer 4</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>new customer</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

\( \pi(i) \) = position of the \( i \)-th label in the ranking

1: MINI  2: Toyota  3: BMW
Label Ranking (more formally)

Given:
- a set of training instances \( \{x_k \mid k = 1 \ldots m\} \subseteq X \)
- a set of labels \( L = \{l_i \mid i = 1 \ldots n\} \)
- for each training instance \( x_k \): a set of pairwise preferences of the form \( l_i \succ_{x_k} l_j \) (for some of the labels)

Find:
- A ranking function \( \mathcal{X} \rightarrow \Omega \) mapping that maps each \( x \in X \) to a ranking \( \succ_x \) of \( L \) (permutation \( \pi_x \)) and generalizes well in terms of a loss function on rankings (e.g., Kendall’s tau)
Existing Approaches

... essentially reduce label ranking to classification:

- Ranking by pairwise comparison
  Fürnkranz and Hüllermeier, ECML-03
- Constraint classification (CC)
  Har-Peled, Roth and Zimak, NIPS-03
- Log linear models for label ranking
  Dekel, Manning and Singer, NIPS-03

  - are efficient but may come with a loss of information
  - may have an improper bias and lack flexibility
  - may produce models that are not easily interpretable
Local Approach (this work)

- Target function $\mathcal{X} \rightarrow \Omega$ is estimated (on demand) in a local way.
- Distribution of rankings is (approx.) constant in a local region.
- Core part is to estimate the locally constant model.
Local Approach  (this work)

- Output (ranking) of an instance $x$ is generated according to a distribution $\mathcal{P}(\cdot | x)$ on $\Omega$.

- This distribution is (approximately) constant within the local region under consideration.

- Nearby preferences are considered as a sample generated by $\mathcal{P}$, which is estimated on the basis of this sample via ML.
Probabilistic Model for Ranking

Mallows model (Mallows, Biometrika, 1957)

\[ P(\sigma|\theta, \pi) = \frac{\exp(-\theta d(\pi, \sigma))}{\phi(\theta, \pi)} \]

with

center ranking \( \pi \in \Omega \)

spread parameter \( \theta > 0 \)

and \( d(\cdot) \) is a right invariant metric on permutations

\[ \forall \pi, \sigma, \nu \in \Omega, \quad d(\pi, \sigma) = d(\pi \nu, \sigma \nu). \]
Inference (complete rankings)

Rankings $\sigma = \{\sigma_1, \ldots, \sigma_k\}$ observed locally.

$$P(\sigma|\theta, \pi) = \prod_{i=1}^{k} P(\sigma_i|\theta, \pi)$$

$$= \prod_{i=1}^{k} \frac{\exp(-\theta d(\sigma_i, \pi))}{\phi(\theta)}$$

$$= \exp\left(-\theta \left(d(\sigma_1, \pi) + \ldots + d(\sigma_k, \pi)\right)\right)$$

$$= \frac{\exp\left(-\theta \sum_{i=1}^{k} d(\sigma_i, \pi)\right)}{\left(\prod_{j=1}^{n} \frac{1-\exp(-j\theta)}{1-\exp(-\theta)}\right)^k}.$$

ML

$$\hat{\pi} = \arg \min_{\pi} \sum_{i=1}^{k} d(\sigma_i, \pi)$$

monotone in $\theta$

$$\frac{1}{k} \sum_{i=1}^{k} d(\sigma_i, \hat{\pi}) = n \exp(-\theta) - \sum_{j=1}^{n} j \exp(-j\theta) \over 1 - \exp(-\theta) - \sum_{j=1}^{n} 1 - \exp(-j\theta)$$
Inference (incomplete rankings)

Probability of an incomplete ranking:

\[ P(E(\sigma_i) \mid \theta, \pi) = \sum_{\sigma \in E(\sigma_i)} P(\sigma \mid \theta, \pi) \]

where \( E(\sigma_i) \) denotes the set of consistent extensions of \( T \).

Example for label set \( \{a,b,c\} \):

<table>
<thead>
<tr>
<th>Observation ( \sigma )</th>
<th>Extensions ( E(\sigma) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a &gt; b )</td>
<td>( a &gt; b &gt; c )</td>
</tr>
<tr>
<td></td>
<td>( a &gt; c &gt; b )</td>
</tr>
<tr>
<td></td>
<td>( c &gt; a &gt; b )</td>
</tr>
</tbody>
</table>
Inference (incomplete rankings) cont.

The corresponding likelihood:

\[ P(\sigma|\theta, \pi) = \prod_{i=1}^{k} P(E(\sigma_i)|\theta, \pi) \]

\[ = \prod_{i=1}^{k} \sum_{\sigma \in E(\sigma_i)} P(\sigma|\theta, \pi) \]

\[ = \frac{\prod_{i=1}^{k} \sum_{\sigma \in E(\sigma_i)} \exp(-\theta d(\sigma, \pi))}{\left(\prod_{j=1}^{n} \frac{1 - \exp(-j\theta)}{1 - \exp(-\theta)}\right)^k} . \]

Exact MLE \((\hat{\pi}, \hat{\theta}) = \arg\max_{\pi, \theta} P(\sigma|\theta, \pi)\) becomes infeasible when \(n\) is large. Approximation is needed.
Inference (incomplete rankings) cont.

Approximation via a variant of EM, viewing the non-observed labels as hidden variables.

- replace the E-step of EM algorithm with a maximization step (widely used in learning HMM, K-means clustering, etc.)

1. Start with an initial center ranking (via generalized Borda count)
2. Replace an incomplete observation with its most probable extension (first M-step, can be done efficiently)
3. Obtain MLE as in the complete ranking case (second M-step)
4. Replace the initial center ranking with current estimation
5. Repeat until convergence
Inference

Not only the estimated ranking $\hat{\pi}$ is of interest ... 

... but also the spread parameter $\hat{\theta}$, which is a measure of precision and, therefore, reflects the confidence/reliability of the prediction (just like the variance of an estimated mean).

The bigger $\hat{\theta}$, the more peaked the distribution around the center ranking.
Label Ranking Trees

Major modifications:

- **split criterion**
  
  Split ranking set $T$ into $T^+$ and $T^-$, maximizing goodness-of-fit
  
  $$
  \frac{|T^+| \cdot \theta^+ + |T^-| \cdot \theta^-}{|T|}
  $$

- **stopping criterion for partition**
  
  1. tree is pure
     any two labels in two different rankings have the same order
  
  2. number of labels in a node is too small
     prevent an excessive fragmentation
Label Ranking Trees

Labels: BMW, Mini, Toyota
## Experimental Results

<table>
<thead>
<tr>
<th></th>
<th>complete rankings</th>
<th>30% missing labels</th>
<th>60% missing labels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CC</td>
<td>IBLR</td>
<td>LRT</td>
</tr>
<tr>
<td>authorship</td>
<td>.920(2)</td>
<td>.936(1)</td>
<td>.882(3)</td>
</tr>
<tr>
<td>bodyfat</td>
<td>.281(1)</td>
<td>.248(2)</td>
<td>.117(3)</td>
</tr>
<tr>
<td>calhousing</td>
<td>.250(3)</td>
<td>.351(1)</td>
<td>.324(2)</td>
</tr>
<tr>
<td>cpu-small</td>
<td>.475(2)</td>
<td>.506(1)</td>
<td>.447(3)</td>
</tr>
<tr>
<td>elevators</td>
<td>.768(1)</td>
<td>.733(3)</td>
<td>.760(2)</td>
</tr>
<tr>
<td>fried</td>
<td>.999(1)</td>
<td>.935(2)</td>
<td>.890(3)</td>
</tr>
<tr>
<td>glass</td>
<td>.846(3)</td>
<td>.865(2)</td>
<td>.883(1)</td>
</tr>
<tr>
<td>housing</td>
<td>.660(3)</td>
<td>.745(2)</td>
<td>.797(1)</td>
</tr>
<tr>
<td>iris</td>
<td>.836(3)</td>
<td>.966(1)</td>
<td>.947(2)</td>
</tr>
<tr>
<td>pendigits</td>
<td>.903(3)</td>
<td>.944(1)</td>
<td>.935(2)</td>
</tr>
<tr>
<td>segment</td>
<td>.914(3)</td>
<td>.959(1)</td>
<td>.949(2)</td>
</tr>
<tr>
<td>stock</td>
<td>.737(3)</td>
<td>.927(1)</td>
<td>.895(2)</td>
</tr>
<tr>
<td>vehicle</td>
<td>.855(2)</td>
<td>.862(1)</td>
<td>.827(3)</td>
</tr>
<tr>
<td>vowel</td>
<td>.623(3)</td>
<td>.900(1)</td>
<td>.794(2)</td>
</tr>
<tr>
<td>wine</td>
<td>.933(2)</td>
<td>.949(1)</td>
<td>.882(3)</td>
</tr>
<tr>
<td>wisconsin</td>
<td>.629(1)</td>
<td>.506(2)</td>
<td>.343(3)</td>
</tr>
<tr>
<td>average rank</td>
<td>2.25</td>
<td>1.44</td>
<td>2.31</td>
</tr>
</tbody>
</table>

**IBLR**: instance-based label ranking  
**LRT**: label ranking trees  
**CC**: constraint classification  
Performance in terms of *Kentall’s tau*
Accuracy (Kendall’s tau)

Typical “learning curves”:

Main observation: Local methods are more flexible and can exploit more preference information compared with the model-based approach.
Take-away Messages

- An instance-based method for label ranking using a probabilistic model.
- Suitable for complete and incomplete rankings.
- Comes with a natural measure of the reliability of a prediction. Makes other types of learners possible: label ranking trees.
  - More efficient inference for the incomplete case.
  - Dealing with variants of the label ranking problem, such as calibrated label ranking and multi-label classification.
Thanks!

Google “kebi germany” for more info.