Trimming the complexity of Ranking by Pairwise Comparison

Samuel Hiard
University of Liège
11th February 2013
Outline

• 1\textsuperscript{st} Part
  – Context and definitions
    • Artificial Intelligence
    • Supervised Learning
    • Label Ranking
  – Problem setting
  – Contribution

• 2\textsuperscript{nd} Part
  – Methods and algorithms
  – Empirical validation
  – Conclusion
Artificial intelligence

• Hard to define because of the subjectivity of the word “intelligence”.

• Literature:
  – AI produces an intelligent response to a given task (Encyclopedia of Artificial Intelligence)
  – AI simulates the behaviour of a human brain (Ibid)
  – “The building of computer programs which perform tasks which are, for the moment, performed in a more satisfactory way by humans because they require high-level mental processes such as: perception learning, memory organization and critical reasoning” (Marvin Lee Minsky, ~1956)

• AI should also be “adaptive”: be able to change its behaviour if the objective or the environment changes
Applications of AI

Problem solving (e.g. Rush Hour)
Applications of AI

Problem solving (e.g. Rush Hour)
Applications of AI

Problem solving (e.g. Rush Hour)
Applications of AI

Problem solving (e.g. Rush Hour)
Applications of AI

Problem solving (e.g. Rush Hour)
Applications of AI

Problem solving (e.g. Rush Hour)

Solved by exploring the possibilities
Applications of AI

Problem solving (e.g. Rush Hour)

• Produces an intelligent response
• Is inspired by the human cognitive behaviour

But

• Requires prior knowledge
  – Exploring every state → solution discovery
  – Rules of the game (legal moves, transition graph)
• Many reachable states → long computational time
Applications of AI

Other approach:
- Let good players solve these puzzles
- Observe and memorize their actions

<table>
<thead>
<tr>
<th>Red car</th>
<th>Blue car</th>
<th>Yellow truck</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>[3,3,HOR]</td>
<td>[5,4,HOR]</td>
<td>[4,6,VERT]</td>
<td>[Blue car, left, 2]</td>
</tr>
<tr>
<td>[3,4,HOR]</td>
<td>[-,-,-]</td>
<td>[2,6,VERT]</td>
<td>[Yellow car, down, 2]</td>
</tr>
<tr>
<td>[3,2,HOR]</td>
<td>[2,1,VERT]</td>
<td>[-,-,-]</td>
<td>[Red car, right, 1]</td>
</tr>
<tr>
<td>[3,2,HOR]</td>
<td>[2,1,VERT]</td>
<td>[-,-,-]</td>
<td>[Blue car, up, 1]</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Solving the problem:

- Find the current situation in the database
- Apply the most frequent move
  - The solver acts like a good player.
  - Trading accuracy for computation reduction

- What if the current situation is *not* in the database?
  - Supervised Learning
Supervised Learning

• Problem: Raw data can be hard to exploit
• Example: Attractivity of long coats

<table>
<thead>
<tr>
<th>Age</th>
<th>Size</th>
<th>Liking</th>
</tr>
</thead>
<tbody>
<tr>
<td>39</td>
<td>1.75</td>
<td>LIKES</td>
</tr>
<tr>
<td>42</td>
<td>1.65</td>
<td>DISLIKES</td>
</tr>
<tr>
<td>27</td>
<td>1.82</td>
<td>DISLIKES</td>
</tr>
<tr>
<td>37</td>
<td>1.77</td>
<td>LIKES</td>
</tr>
</tbody>
</table>

• Would a 35-years old, 1.73m height man like a long coat?
Supervised Learning

• Problem: Raw data can be hard to exploit
• Example: Attractivity of long coats

<table>
<thead>
<tr>
<th>Age</th>
<th>Size</th>
<th>Liking (/10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>39</td>
<td>1.75</td>
<td>8</td>
</tr>
<tr>
<td>42</td>
<td>1.65</td>
<td>3</td>
</tr>
<tr>
<td>27</td>
<td>1.82</td>
<td>2</td>
</tr>
<tr>
<td>37</td>
<td>1.77</td>
<td>9</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

• How much would a 35-years old, 1.73m height man like a long coat?
Supervised Learning

- **Objective: Model construction**

<table>
<thead>
<tr>
<th>Age</th>
<th>Size</th>
<th>Liking</th>
</tr>
</thead>
<tbody>
<tr>
<td>39</td>
<td>1.75</td>
<td>LIKES</td>
</tr>
<tr>
<td>42</td>
<td>1.65</td>
<td>DISLIKES</td>
</tr>
<tr>
<td>27</td>
<td>1.82</td>
<td>DISLIKES</td>
</tr>
<tr>
<td>37</td>
<td>1.77</td>
<td>LIKES</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

New input $x'$

Model

Predicted output $y'$
Supervised Learning

- Premise: Objects with the same class tend to cluster
Model construction

• Example: c4.5 algorithm

![Graph with age and height data points for model construction example.]
Example: c4.5 algorithm

Age >= 31?

Upper objects

DISLIKES
Model construction

- Example: c4.5 algorithm

```
<table>
<thead>
<tr>
<th>Age &gt;= 31?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height &lt; 1.68?</td>
</tr>
<tr>
<td>DISLIKES</td>
</tr>
<tr>
<td>LIKES</td>
</tr>
</tbody>
</table>
```

11th February 2013  19  Samuel Hiard
Model prediction

• Prediction for (possibly) unseen objects

- A 35-year old, 1.73m height man will thus like a long coat
Model prediction

- Prediction for border objects can be problematic
- Fitting the sample ≠ Representing the general rule
- We need to evaluate our model
Model evaluation

- Test on new data (resampling, partitioning)

Accuracy = \frac{\text{#well classified}}{\text{#total predictions}}

In this example,
accuracy = \frac{21}{25} = 84\%
# Different types of goal

<table>
<thead>
<tr>
<th>Goal type (y)</th>
<th>Problem type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 symbolic value</td>
<td>Classification</td>
<td>y = « DISLIKES »</td>
</tr>
<tr>
<td>1 numerical value</td>
<td>Regression</td>
<td>y = 12.34</td>
</tr>
<tr>
<td>1 image</td>
<td>Image prediction</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Image completion</td>
<td>y = <img src="image.png" alt="Image" /></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>An ordered subset of symbolic values</td>
<td>Label Ranking</td>
<td>y = <img src="image.png" alt="Image" /> &gt; <img src="image.png" alt="Image" /> &gt; <img src="image.png" alt="Image" /></td>
</tr>
</tbody>
</table>
Label Ranking

- Particular case of Supervised Learning
- The output is not a number nor a symbol, but an order between labels (« ranking »), e.g.
  \[ A > B > C > D \]  \[ A > B > C \]
  (total order) (partial order, more frequent in practice)

- Provides more information than a single label.
  E.g.: A book store (say Amazon) can perform recommendations to increase its sales rate.
Recommender systems

<table>
<thead>
<tr>
<th>ID</th>
<th>Age</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>32</td>
<td>1.68</td>
</tr>
</tbody>
</table>

New user

Is submitted to

Model

Are used to build

User ratings

<table>
<thead>
<tr>
<th>Age</th>
<th>Size</th>
<th>Preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>1.62</td>
<td>a &gt; b &gt; d &gt; c</td>
</tr>
<tr>
<td>37</td>
<td>1.75</td>
<td>b &gt; d &gt; c</td>
</tr>
<tr>
<td>18</td>
<td>1.83</td>
<td>e &gt; d &gt; a &gt; b</td>
</tr>
</tbody>
</table>

Predicted ranking

Ranking

a > c > d > e > b
Recommender systems

New user

User purchases

User ratings

Are used to build

Model

User ratings

Are used to build

Predicted ranking

Is filtered by

Recommended ranking

ID | Age | Size
---|-----|-----
12 | 32  | 1.68

ID | Bought
---|-----
12 | a, c
...
...

ID | Bought
---|-----
11th February 2013 | 26
Label Ranking

• Could be solved by standard SL algorithms, by attributing a number to each possible permutation (e.g. lexicographic order), but:
  – Would be inefficient with partial rankings
  – Would not take advantage of the structure, e.g.:

    \[
    \begin{align*}
    \text{Single label displacement} & \quad \{ A > B > C > D > E \rightarrow 1 \\
    & \quad \{ E > A > B > C > D \rightarrow 97 \\
    & \quad \{ D > E > C > B > A \rightarrow 96 \\
    \end{align*}
    \]

  \rightarrow \text{Algorithms specific to label ranking}
Label Ranking

• Decompose the problem
  – Learn a utility function per label,
    « How much would you rate this item? »
  e.g.:

  • $U(\text{kiwi}) = 2$, $U(\text{banana}) = 1$, $U(\text{apple}) = 3$,

  $\Rightarrow \text{apple} > \text{kiwi} > \text{banana}$
Label Ranking

• Decompose the problem
  – Learn a model for each pair of labels,
    « Would you rate item A higher than item B? »
  e.g.:

  • Apple > Kiwi and Apple > Banana and Kiwi > Banana

  → Apple > Kiwi > Banana

Strategy used in this work (RPC)
Ranking by Pairwise Comparison (RPC)

Eyke Hüllermeier and Johannes Fürnkranz, 2004

• Step 1: Decomposition into pairwise comparisons

Original data

\[ x^i = (\text{age, size, } \ldots) \text{ for } i^{th} \text{ object} \]

<table>
<thead>
<tr>
<th>Input</th>
<th>Preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x^1 )</td>
<td>( a &gt; b &gt; c )</td>
</tr>
<tr>
<td>( x^2 )</td>
<td>( b &gt; c &gt; a )</td>
</tr>
<tr>
<td>( x^3 )</td>
<td>( c &gt; b &gt; a )</td>
</tr>
<tr>
<td>( x^4 )</td>
<td>( a &gt; c )</td>
</tr>
</tbody>
</table>

11th February 2013 30 Samuel Hiard
Ranking by Pairwise Comparison (RPC)

Eyke Hüllermeier and Johannes Fürnkranz, 2004

- Step 2: Training of all pairwise comparators

<table>
<thead>
<tr>
<th>Input</th>
<th>Preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x^1$</td>
<td>true</td>
</tr>
<tr>
<td>$x^2$</td>
<td>false</td>
</tr>
<tr>
<td>$x^3$</td>
<td>false</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input</th>
<th>Preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x^3$</td>
<td>true</td>
</tr>
<tr>
<td>$x^2$</td>
<td>false</td>
</tr>
<tr>
<td>$x^3$</td>
<td>false</td>
</tr>
<tr>
<td>$x^4$</td>
<td>true</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input</th>
<th>Preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x^1$</td>
<td>true</td>
</tr>
<tr>
<td>$x^2$</td>
<td>true</td>
</tr>
<tr>
<td>$x^3$</td>
<td>false</td>
</tr>
</tbody>
</table>

11th February 2013

Samuel Hiard

31
Ranking by Pairwise Comparison (RPC)

Eyke Hüllermeier and Johannes Fürnkranz, 2004

- **Prediction**: Aggregating comparator outputs
• Needs $O(N^2)$ in terms of number $N$ of labels
• Computational and Space complexity
• Non applicable in practice to huge datasets
• Intuition:
  – There might be some redundant and/or noisy comparisons in this set of \(O(N^2)\) comparisons (no theoretical evidence)
  – Reducing this set would improve the speed performance of RPC and could be (nearly) as accurate

• Contribution:
  – The design of algorithms aiming at selecting the optimal subset of \(T\) comparisons and the study of the behaviour of a RPC scheme in terms of accuracy vs. number \(T\) of comparisons.
Contribution

Database → LS → N(N-1)/2 comparisons → Model construction → Comparator set evaluation → Accuracy measure
Contribution

Database → LS → N(N-1)/2 comparisons → Comparator set evaluation

LS → N(N-1)/2 comparisons → Model construction

TS → N(N-1)/2 comparisons → Comparator set evaluation

Model construction → Comparison set selection algorithms (T comparisons)

Comparator set evaluation → Accuracy measure
Outline

• 1\textsuperscript{st} Part
  – Context and definitions
  – Problem setting
  – Contribution

• 2\textsuperscript{nd} Part
  – Methods and algorithms
    • Subset selection methods (PR, EDA, EGS, RGS)
    • Evaluating a prediction and a subset of comparisons
    • Base learner: Extra-trees
  – Empirical validation
    • Databases
    • Results and parameters discussion
    • Complexity reduction
  – Conclusion
Guideline

Database → LS → Comparator set evaluation → Accuracy measure

Database → TS → Model construction

N(N-1)/2 comparisons

Comparison set selection algorithms (T comparisons)
Pure random selection (PR)

- No hard computations
- No need to store the LS outputs
  → Baseline

Complexity (\# subset evaluations) : 0

All available comparisons
Random uniform selection
Subset of size $T$
Estimation of distribution algorithm (EDA)

- $P(q) = \text{uniform}$
- Draw $s$ sets using $P(q)$
- Evaluate the sets (Spearman)
- Update $P(q)$ using the content of these $b$ sets
- Keep the best $b$ sets

Repeat $j$ times

Output: Best set across all iterations

Complexity: $sj$

11th February 2013

Samuel Hiard
**« Exhaustive » Greedy Search (EGS)**

1. Draw a set using $P(q)$
2. Iteratively replace each comparison by its best alternative across unselected comparisons
3. Evaluate the set
4. Convergence?
   - yes
   - no

Output this set

**Complexity:** $jT \mathcal{O}(N^2)$
Randomized Greedy Search (RGS)

1. Draw a set using $P(q)$
2. Randomly select two comparisons: $q$ (in the set) and $q'$ (not in the set)
3. Replace $q'$ by $q$ in the set and evaluate it

- Improvement?
  - Yes: Keep the set unchanged
    - Decrease $P(q)$
    - Renormalize $P$
  - No: Reverse the swap
    - Decrease $P(q')$
    - Renormalize $P$

4. STOP? $(j_1, j_2, j_3)$
   - Yes: Output this set
   - No: Repeat

Complexity: $\frac{j_1 + j_3}{2}$
Guideline

- **N(N-1)/2 comparisons**
- **Comparison set selection algorithms** (T comparisons)
- **Model construction**
- **Comparator set evaluation**
- **Accuracy measure**
Guideline

N(N-1)/2 comparisons

Comparison

Resampling/ Updating

Comparison set evaluation

Model construction

Comparator set evaluation

Accuracy measure
• 0/1 Loss

\[ L = \sum_{i=1}^{\#TS} \begin{cases} 1 & \hat{y}^i \neq y^i \\ 0 & \hat{y}^i = y^i \end{cases} \]

\( y^i = \) true output of object \( i \)

\( \hat{y}^i = \) predicted output for object \( i \)

• Least squares error

\[ L = \sum_{i=1}^{\#TS} (\hat{y}^i - y^i)^2 \]
Computing the similarity between two rankings

- **Spearman’s rank correlation coefficient**

\[
(y, \hat{y}) \rightarrow \rho_S = 1 - \frac{6 \sum_{k=0}^{\#y} (y_k - \hat{y}_k)^2}{\# y((\# y)^2 - 1)}
\]

\# y = Number of labels in ranking \(y\)

\(y_k = \) Position of label \(k\) in ranking \(y\)

- **Kendall Tau rank correlation coefficient**

\[
(y, \hat{y}) \rightarrow \rho_K = 1 - \frac{4\#\{(k, l) \mid (k < l) \land (y_k > y_l) \land (\hat{y}_k < \hat{y}_l)\}}{\# y(\# y - 1)}
\]
• Overall ranking score $S_{TS}$ on the sample $TS$

$$S_{TS} = \frac{\sum_{i=1}^{#TS} \rho(\hat{y}^i, y^i)}{#TS}$$
Evaluating a set of comparisons on a sample

Default order = a > b > c

<table>
<thead>
<tr>
<th>Input</th>
<th>Supervision</th>
</tr>
</thead>
<tbody>
<tr>
<td>x¹</td>
<td>a &gt; b &gt; c</td>
</tr>
<tr>
<td>x²</td>
<td>b &gt; c &gt; a</td>
</tr>
<tr>
<td>x³</td>
<td>c &gt; b &gt; a</td>
</tr>
<tr>
<td>x⁴</td>
<td>a &gt; c</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Spearman</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>-1</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ordered Subset</th>
</tr>
</thead>
<tbody>
<tr>
<td>a &gt; c &gt; b</td>
</tr>
<tr>
<td>b &gt; c &gt; a</td>
</tr>
<tr>
<td>a &gt; b &gt; c</td>
</tr>
<tr>
<td>a &gt; c</td>
</tr>
</tbody>
</table>

Overall ranking score $S_{TS} = (0.5+1-1+1)/4 = 0.375$

Optimization phase: another computational issue
Evaluating a set of comparisons during the optimization phase

• $N = 4$, $\#\text{Comp} = 6$, $T = 3$, RGS

Models

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Evaluating a set of comparisons during the optimization phase

- $N = 4$, $\#\text{Comp} = 6$, $T = 3$, RGS

<table>
<thead>
<tr>
<th>Models</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

- Iteration 1: Comparisons = \{1, 3, 4\}
Evaluating a set of comparisons during the optimization phase

- $N = 4$, $\#\text{Comp} = 6$, $T = 3$, RGS

<table>
<thead>
<tr>
<th>Models</th>
<th>No</th>
<th>Yes</th>
<th>Yes</th>
<th>Yes</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

- Iteration 1: Comparisons = $\{1, 3, 4\}$
- Iteration 2: Comparisons = $\{2, 3, 4\}$
Evaluating a set of comparisons during the optimization phase

• N = 4, #Comp = 6, T = 3, RGS

<table>
<thead>
<tr>
<th>Models</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

• Iteration 1: Comparisons = {1, 3, 4}
• Iteration 2: Comparisons = {2, 3, 4}
• ...
• Iteration j₃: Comparisons = {0, 3, 5}
Evaluating a set of comparisons during the optimization phase

- \( N = 4, \ #\text{Comp} = 6, \ T = 3, \ RGS \)

<table>
<thead>
<tr>
<th>Models</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Iteration 1: Comparisons = \{1, 3, 4\}
- Iteration 2: Comparisons = \{2, 3, 4\}
- ...
- Iteration \( j_3 \): Comparisons = \{0, 3, 5\}

- Each model was built and stored.
  
  Same complexity issue as RPC.
Evaluating a set of comparisons on the LS

Default order = a > b > c

<table>
<thead>
<tr>
<th>Input</th>
<th>Preference</th>
<th>Input</th>
<th>Preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>x₁</td>
<td>true</td>
<td>x₁</td>
<td>true</td>
</tr>
<tr>
<td>x₂</td>
<td>false</td>
<td>x₂</td>
<td>false</td>
</tr>
<tr>
<td>x₃</td>
<td>false</td>
<td>x₃</td>
<td>false</td>
</tr>
<tr>
<td>x₄</td>
<td>true</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input</th>
<th>Supervision</th>
<th>Input</th>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>x₁</td>
<td>a &gt; b &gt; c</td>
<td>x₁</td>
<td>a &gt; b &gt; c</td>
</tr>
<tr>
<td>x₂</td>
<td>b &gt; c &gt; a</td>
<td>x₂</td>
<td>b &gt; c &gt; a</td>
</tr>
<tr>
<td>x₃</td>
<td>c &gt; b &gt; a</td>
<td>x₃</td>
<td>b &gt; c &gt; a</td>
</tr>
<tr>
<td>x₄</td>
<td>a &gt; c</td>
<td>x₄</td>
<td>a &gt; b &gt; c</td>
</tr>
</tbody>
</table>

Spearman

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Ordered Subset

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a &gt; b &gt; c</td>
<td>b &gt; c &gt; a</td>
<td>b &gt; c &gt; a</td>
</tr>
<tr>
<td>a &gt; c</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Overall ranking score = (1+1+0.5+1)/4 = 0.875
Guideline

- **Database**
- **LS**
- **TS**

**Model construction**

- **N(N-1)/2 comparisons**
- **Comparator set evaluation**

**Comparison set selection algorithms**

(T comparisons)

**Accuracy measure**
Extremely Randomized Trees

Decision tree:

Age >= 31?

- yes
  - Height < 1.68?
    - yes: DISLIKES
    - no: LIKES

- no: DISLIKES

<table>
<thead>
<tr>
<th>Age</th>
<th>Size</th>
<th>Liking</th>
</tr>
</thead>
<tbody>
<tr>
<td>39</td>
<td>1.75</td>
<td>LIKES</td>
</tr>
<tr>
<td>42</td>
<td>1.65</td>
<td>DISLIKES</td>
</tr>
<tr>
<td>27</td>
<td>1.82</td>
<td>DISLIKES</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
• Decision trees: The node split which minimizes the entropy is chosen (deterministic)

![Graph showing decision tree splits based on age and height]
• Extra-trees: $K$ couples feature/threshold are randomly selected as node split candidates. The best node split among these $K$ candidates is chosen (added randomization)
• $M$ randomized trees are built, and their predictions are aggregated.
• **Classification**: Each tree outputs the majority class of the reached leaf. The majority class among all $M$ predictions is the final result.
• **Regression**: Each tree outputs the mean value of the labels in the reached leaf. The mean value of all $M$ predictions is the final result.
Extremely Randomized Trees

Parameters:

- $n_{\text{min}}$: Number of objects for node split, pruning control
  - $n_{\text{min}} = 2$ for classification (fully grown)
  - $n_{\text{min}} = 5$ for regression

- $K$: Number of couples feature/threshold, output dependence
  - $K = \sqrt{\#\text{Attributes}}$ for classification
  - $K = \#\text{Attributes}$ for regression

- $M$: Number of trees, computational effort control
  - No default values. Accuracy vs Computational time
Advantages:

• Strongly reduces the variance with no significant increase in bias
• Very fast

Inconvenients:

• May require fine-tuning the parameters (although standard values are often optimal)
Outline

• 1\textsuperscript{st} Part
  – Context and definitions
  – Problem setting
  – Contribution

• 2\textsuperscript{nd} Part
  – Methods and algorithms
    • Subset selection methods (PR, EDA, EGS, RGS)
    • Evaluating a prediction and a subset of comparisons
    • Base learner: Extra-trees
  – Empirical validation
    Databases
      • Results and parameters discussion
      • Complexity reduction
  – Conclusion
Used databases

- OMIB, synthetic, stability of power system
  ![Database OMIB Diagram](image)

- Sushi, real life, user preference over sushi’s
  ![Sushi Preference](image)

- MovieLens, real life, user preference over movies
  ![Movie Preferences](image)
OMIB

7 numerical attributes

1 numerical output

<table>
<thead>
<tr>
<th>Object ID</th>
<th>PL</th>
<th>PU</th>
<th>QU</th>
<th>VINF</th>
<th>VL</th>
<th>XINF</th>
<th>CCT-SBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-97.479</td>
<td>950.889</td>
<td>-6.4902</td>
<td>1.00208</td>
<td>0.998238</td>
<td>81.9069</td>
<td>0.1265</td>
</tr>
<tr>
<td>2</td>
<td>-90.052</td>
<td>704.46</td>
<td>888.322</td>
<td>1.04706</td>
<td>1.038</td>
<td>77.6292</td>
<td>0.3687</td>
</tr>
<tr>
<td>3</td>
<td>-68.203</td>
<td>1038.83</td>
<td>-550.47</td>
<td>1.10339</td>
<td>1.10283</td>
<td>51.0616</td>
<td>0.1792</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>20,000</td>
<td>-101.76</td>
<td>1073.68</td>
<td>700.649</td>
<td>1.07136</td>
<td>1.08313</td>
<td>46.2877</td>
<td>0.2632</td>
</tr>
</tbody>
</table>

CCT-SBS: 0.18

Corresponding ordering: 3 > 4 > 2 > 5 > 1 > 6 > 0 > 7 > 8 > 9
Sushi

11 attributes

<table>
<thead>
<tr>
<th>User ID</th>
<th>Gender</th>
<th>Age</th>
<th>Time ToFill</th>
<th>MP</th>
<th>MR</th>
<th>MEW</th>
<th>...</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>8,270</td>
<td>M</td>
<td>1</td>
<td>219</td>
<td>13</td>
<td>3</td>
<td>E</td>
<td>...</td>
<td>10,20,13,9,16,23,30,17,18,91</td>
</tr>
<tr>
<td>8,053</td>
<td>M</td>
<td>3</td>
<td>259</td>
<td>24</td>
<td>5</td>
<td>E</td>
<td>...</td>
<td>43,47,9,27,10,66,5,18,21,42</td>
</tr>
<tr>
<td>6,823</td>
<td>F</td>
<td>3</td>
<td>249</td>
<td>30</td>
<td>7</td>
<td>W</td>
<td>...</td>
<td>15,2,19,16,55,46,95,32,3,18</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>8,756</td>
<td>M</td>
<td>3</td>
<td>316</td>
<td>13</td>
<td>3</td>
<td>E</td>
<td>...</td>
<td>8,1,76,0,2,5,32,4,35,80</td>
</tr>
</tbody>
</table>

1 ranking (10 out of 100)

500 objects

5000 objects

Random selection

Clustering

10 clusters

Narrow variance

Sushi_0

Sushi_3

11th February 2013

Samuel Hiard
### MovieLens

**5 attributes**

<table>
<thead>
<tr>
<th>User ID</th>
<th>Age</th>
<th>Gender</th>
<th>Occupation</th>
<th>Zip code</th>
<th>Movie ID</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24</td>
<td>M</td>
<td>0</td>
<td>85711</td>
<td>46</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>24</td>
<td>M</td>
<td>0</td>
<td>85711</td>
<td>132</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>53</td>
<td>F</td>
<td>1</td>
<td>94043</td>
<td>35</td>
<td>5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>943</td>
<td>22</td>
<td>M</td>
<td>5</td>
<td>77841</td>
<td>1534</td>
<td>1</td>
</tr>
</tbody>
</table>

**1 numerical**

- 100,000 ratings
- 943 objects

**Input:**

- (User1, Movie16, 3)
- (User1, Movie235, 5)
- (User1, Movie527, 2)
- (User1, Movie988, 3)

**Corresponding ordering:**

235 > 16 > 988 > 527

(20 → 736) out of 1682
• The data is so sparse that some comparisons cannot be modeled, e.g.

<table>
<thead>
<tr>
<th>Objects in the learning sample (x,y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x₁, a &gt; c)</td>
</tr>
<tr>
<td>(x₂, b &gt; c &gt; d)</td>
</tr>
<tr>
<td>(x₃, a &gt; d &gt; c)</td>
</tr>
</tbody>
</table>

How do we infer the preference relation between « a » and « b »?
Sparsity issue

• Using a dummy model (UDM)

\[
\text{Any input } x \quad \rightarrow \quad \text{Dummy model} \quad \rightarrow \quad 0.5
\]

This comparator will thus provide half a point to both the labels that it compares.
### Sparsity issue

- **Dropping empty comparison (DEC)**

<table>
<thead>
<tr>
<th></th>
<th>a &gt; b</th>
<th>a &gt; c</th>
<th>a &gt; d</th>
<th>a &gt; e</th>
<th>b &gt; c</th>
<th>b &gt; d</th>
<th>b &gt; e</th>
<th>c &gt; d</th>
<th>c &gt; e</th>
<th>d &gt; e</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>-</td>
<td>true</td>
<td>false</td>
<td>-</td>
<td>false</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>-</td>
<td>false</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>true</td>
<td>true</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>-</td>
<td>true</td>
<td>false</td>
<td>-</td>
<td>false</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>-</td>
<td>true</td>
<td>true</td>
<td>-</td>
<td>true</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

All objects in LS

|        | 15    | 6     | 3     | 0     | 4     | 9     | 1     | 3     | 0     | 1     |
## Sparsity issue

- **Dropping empty comparison (DEC)**

<table>
<thead>
<tr>
<th>a &gt; b</th>
<th>a &gt; c</th>
<th>a &gt; d</th>
<th>a &gt; e</th>
<th>b &gt; c</th>
<th>b &gt; d</th>
<th>b &gt; e</th>
<th>c &gt; d</th>
<th>c &gt; e</th>
<th>d &gt; e</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>All objects in LS</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 6 3 0 4 9 1 3 0 1</td>
</tr>
</tbody>
</table>
Sparsity issue

- Dropping empty comparison (DEC)

<table>
<thead>
<tr>
<th>a &gt; b</th>
<th>a &gt; c</th>
<th>a &gt; d</th>
<th>a &gt; e</th>
<th>b &gt; c</th>
<th>b &gt; d</th>
<th>b &gt; e</th>
<th>c &gt; d</th>
<th>c &gt; e</th>
<th>d &gt; e</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>-</td>
<td>true</td>
<td>false</td>
<td>-</td>
<td>false</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>-</td>
<td>-</td>
<td>false</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>true</td>
<td>true</td>
<td>-</td>
<td>-</td>
<td>true</td>
<td>-</td>
</tr>
<tr>
<td>true</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>-</td>
<td>true</td>
<td>false</td>
<td>-</td>
<td>false</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>-</td>
<td>true</td>
<td>true</td>
<td>-</td>
<td>true</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

15 6 3 0 4 9 1 3 0 1
### Sparsity issue

- **Dropping empty comparison (DEC)**

<table>
<thead>
<tr>
<th></th>
<th>a &gt; b</th>
<th>a &gt; c</th>
<th>a &gt; d</th>
<th>a &gt; e</th>
<th>b &gt; c</th>
<th>b &gt; d</th>
<th>b &gt; e</th>
<th>c &gt; d</th>
<th>c &gt; e</th>
<th>d &gt; e</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>-</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>-</td>
<td>false</td>
<td>-</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>-</td>
<td>false</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>false</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>true</td>
<td>true</td>
<td>-</td>
<td>true</td>
<td>-</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>-</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>-</td>
<td>false</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>-</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>-</td>
<td>true</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

| All objects in LS | 15 | 6 | 3 | 0 | 4 | 9 | 1 | 3 | 0 | 1 |
Outline

• 1st Part
  – Context and definitions
  – Problem setting
  – Contribution

• 2nd Part
  – Methods and algorithms
    • Subset selection methods (PR, EDA, EGS, RGS)
    • Evaluating a prediction and a subset of comparisons
    • Base learner: Extra-trees
  – Empirical validation
    • Databases
  Results and parameters discussion
    • Complexity reduction
  – Conclusion
Accuracy figures

![Diagram showing accuracy figures with box plots for State-of-the-art and RPC, and Spearman's rho values ranging from -0.6 to 0.4. The x-axis represents the number of used comparators ranging from 10 to 1000. The y-axis shows Spearman's rho values with Median, Max, Min, and Outlier values indicated.]

11th February 2013

Samuel Hiard
Accuracy on Omib (Classification)

EGS and EDA are the most efficient.
Accuracy on Sushi_0 (Classification, DEC)

RGS is the most efficient
Accuracy on Sushi_3 (Classification, DEC)

RGS is the most efficient
Accuracy on MovieLens (Classification, DEC)

RGS is the most efficient
Pre-trimming the comparisons which have few examples

(Sushi_3, RGS, Regression mode)

- RPC’s performance is decreased
- RGS converges faster
Robustness w.r.t. tree parameters

- Varying one parameter at a time and keeping the standard selection for the remaining ones (Omib, RGS, Classification mode)
- The tree parameters have slim to none influence on the outcome
Accuracy on Omib (Clas vs. Regr)

Regression improves the predictions

Accuracy on TS with PR method

Accuracy on TS with EGS method

11th February 2013
Samuel Hiard
Computing the mean order

For each given ranking, we attribute

\[
\frac{(p - i + 1)(N + 1)}{(p + 1)} \quad \text{votes to each label on rank } i \in \{1 \ldots p\}, \text{ and}
\]

\[
\frac{(N + 1)}{2} \quad \text{votes to each missing label}
\]

(W. Cheng and E. Hüllermeier, A new instance-based label ranking approach using the mallows model, 2009)
Computing the mean order

Example: $N = 99$
Ranking = $L_{53} > L_{12} > L_7 > L_{11} > L_{85} > L_0 > L_{46} > L_{91} > L_{22}$ (9 labels)

<table>
<thead>
<tr>
<th>Label</th>
<th>Votes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_{53}$</td>
<td>90</td>
</tr>
<tr>
<td>$L_{12}$</td>
<td>80</td>
</tr>
<tr>
<td>$L_7$</td>
<td>70</td>
</tr>
<tr>
<td>$L_{11}$</td>
<td>60</td>
</tr>
<tr>
<td>$L_{85}$</td>
<td>50</td>
</tr>
<tr>
<td>$L_0$</td>
<td>40</td>
</tr>
<tr>
<td>$L_{46}$</td>
<td>30</td>
</tr>
<tr>
<td>$L_{91}$</td>
<td>20</td>
</tr>
<tr>
<td>$L_{22}$</td>
<td>10</td>
</tr>
<tr>
<td>All other</td>
<td>50</td>
</tr>
</tbody>
</table>

Every missing label

- Repeat this process for each object in LS
- Sort labels according to total score
Using the mean order to break ties (Omib)

Accuracy on TS with EGS method

Spearman's rho vs Number of used comparators

Regr

Clas

Reverse alphanum

Mean order
Using the mean order to break ties (MovieLens)

Reverse alphanum

Mean order
Outline

• 1\textsuperscript{st} Part
  – Context and definitions
  – Problem setting
  – Contribution

• 2\textsuperscript{nd} Part
  – Methods and algorithms
    • Subset selection methods (PR, EDA, EGS, RGS)
    • Evaluating a prediction and a subset of comparisons
    • Base learner: Extra-trees
  – Empirical validation
    • Databases
    • Results and parameters discussion
    Complexity reduction
  – Conclusion
Complexity gain

- RGS: Subset selection time $<<$ model building time
  $\Rightarrow$ Computational time gain
Summary

• Definitions:
  – Artificial Intelligence
  – Supervised Learning
  – Label Ranking

• Ranking by Pairwise Comparison:
  – Algorithm
  – Evaluation
  – Complexity issue

• Subset selection algorithms:
  – Description
  – Evaluation during optimization proxy
  – Model for base learner

• Validation
  – Databases and sparsity issue
  – Accuracy
  – Complexity
Conclusion

• The RPC algorithm has a complexity burden
• We proposed to reduce the set of $N(N-1)/2$ comparisons to a subset of size $T \approx O(N)$
• We designed several algorithms which aim at finding the optimal subset for a given $T$, prior to building a SL model, RGS being so far the best alternative
• We also proposed an \textit{a priori} reduction based on the number of objects in the comparator’s LS which improves the performance of RGS
• We thus reduced the complexity of RPC without significantly dropping the accuracy, making this approach applicable to problems with a large number $N$ of labels
Future work

• Prior knowledge (ex: Sushi)
• Controlling partial ranking and noise (OMIB)
  – Feature noise
  – Output noise (partial ranking, label displacement)
• Algorithmic improvements
  – Optimizing with a given budget
  – Using the mean order to select $Q$ (subset of $T$ comparisons)
  – Weighted distribution based on LS
Optimizing with a given budget

Comparator 1

Comparator 2

... Comparator T

Tree budget = 500T trees
Using the mean order to select $Q$

- Compute mean order
  - e.g. $5 > 2 > 1 > 6 > 4 > 3$
- Compare labels at position $i$ and $i+1$
  - $5 > 2$
  - $2 > 1$
  - $1 > 6$
  - $6 > 4$
  - $4 > 3$
- Better than PR?
  - Preliminary results (Sushi_3):
    - Classification mode: Always worse than PR (especially when $T$ is large)
    - Regression mode: Always better than PR. Better than RGS if $T \leq 20$
  $\rightarrow$ The top ranked items seems to contain useful information
  $\rightarrow$ Room for improvement
## Weighted distribution based on LS

<table>
<thead>
<tr>
<th>a &gt; b</th>
<th>a &gt; c</th>
<th>a &gt; d</th>
<th>a &gt; e</th>
<th>b &gt; c</th>
<th>b &gt; d</th>
<th>b &gt; e</th>
<th>c &gt; d</th>
<th>c &gt; e</th>
<th>d &gt; e</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>-</td>
<td>true</td>
<td>false</td>
<td>-</td>
<td>false</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>-</td>
<td>-</td>
<td>false</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>true</td>
<td>true</td>
<td>-</td>
<td>-</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>-</td>
<td>true</td>
<td>false</td>
<td>-</td>
<td>false</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>-</td>
<td>true</td>
<td>true</td>
<td>-</td>
<td>true</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#obj</th>
<th>15</th>
<th>6</th>
<th>3</th>
<th>0</th>
<th>4</th>
<th>9</th>
<th>1</th>
<th>3</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
</table>

| P(q)  | 0.37 | 0.14 | 0.07 | 0 | 0.10 | 0.21 | 0.02 | 0.07 | 0 | 0.02 |
Thank you for your attention
Thank you for your attention
Using perfect models

Accuracy on TS with EGS method

OMIB

Accuracy on TS with RGS method

Sushi_0

Accuracy on TS with RGS method

MovieLens

Accuracy on TS with RGS method

Sushi_3
MovieLens: imposing alphanum order

- Votes → Ranking
- Partial ranking (e.g. a>b and c>d)
  - Training the pairwise comparators? OK
  - Spearman evaluation? KO (ground truth unknown)

- Solutions:
  - Evaluating with Kendall Tau (state-of-the-art?)
  - Output = ordered subset of labels
  → Arbitrarily breaking ties