Ranking

Contents

• Ranking Problems
• Comparing Rankings
  • Pairwise instances
  • Kendall’s tau
  • Spearman’s rank order correlation
• Paper ‘Learning to Order Things’
  • Rank orderings
  • Preference functions
  • NP complete problem
• Ranking experts
• Results
The Ranking Problem

Given a set of objects $X$ that are ordered by some preference label, build a function that predicts the rank of new instances.

Applications
- Netflix – predicting user’s future requests
- Ranking documents in order of relevance based on word occurrences
- Ranking emails in order of urgency
- Search engines

‘... obtaining preference information may be easier than obtaining information for classification or regression’ [Cohen et al, Learning to Order Things]

Comparing Rankings

Count the number of examples that are assigned the wrong rank …

<table>
<thead>
<tr>
<th>Films</th>
<th>Correct ranking [ 3 4 5 1 2 ]</th>
<th>Algorithm rank [ 5 4 3 2 1 ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Pairwise instances

<table>
<thead>
<tr>
<th>Is A ranked above B in R1?</th>
<th>Yes.</th>
<th>Discordant pair</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is A ranked above B in R2?</td>
<td>No.</td>
<td></td>
</tr>
<tr>
<td>Is D ranked above C in R1?</td>
<td>Yes.</td>
<td>Concordant pair</td>
</tr>
<tr>
<td>Is D ranked above C in R2?</td>
<td>Yes.</td>
<td></td>
</tr>
</tbody>
</table>
Kendall’s $\tau$

Given two rankings $r_a$ and $r_b$,

$$\tau(r_a, r_b) = \frac{P - Q}{P + Q}$$

$P = \text{number of concordant pairs (rankings agree)}$
$Q = \text{number of discordant pairs (rankings disagree)}$

For similar rankings, $\tau \rightarrow 1$. For dissimilar rankings, $\tau \rightarrow -1$.

Spearman’s rank order correlation

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

$d_i = \text{the difference between each rank of corresponding values of } x \text{ and } y$
$n = \text{the number of pairs of values}.$

The 6 comes from the convergence of the series of the sum of squares:

$$\sum_{i=1}^{n} d_i^2 = \frac{n(n + 1)(2n + 1)}{6}$$

So it plays a normalizing role.

Alternate formulation: Given two rankings $R$ and $S$ ...

$$\theta = \frac{\sum (R_i - \bar{R})(S_i - \bar{S})}{\sqrt{\sum (R_i - \bar{R})^2 \sum (S_i - \bar{S})^2}}$$
Comparing Rankings - Example

Correct ranking [ 3 4 5 1 2 ]
Algorithm rank [ 5 4 3 2 1 ]

Sum of squared rank differences: 4+0+4+1+1=10

<table>
<thead>
<tr>
<th>Conc</th>
<th>Disc</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td>0</td>
</tr>
<tr>
<td>AC</td>
<td>0</td>
</tr>
<tr>
<td>AD</td>
<td>1</td>
</tr>
<tr>
<td>AE</td>
<td>1</td>
</tr>
<tr>
<td>BC</td>
<td>1</td>
</tr>
<tr>
<td>BD</td>
<td>1</td>
</tr>
<tr>
<td>BE</td>
<td>1</td>
</tr>
<tr>
<td>CD</td>
<td>1</td>
</tr>
<tr>
<td>CE</td>
<td>1</td>
</tr>
<tr>
<td>DE</td>
<td>0</td>
</tr>
</tbody>
</table>

Kendalls $\tau = \frac{7-3}{7+3} = 0.4$

Spearmans $\rho = 1 - \frac{6 \cdot 10}{5 \cdot (25-1)} = 0.5$

Online Ranking

What is online learning?

Process data as soon as it is received, learn, and then predict on the next example. Update your learning function in response to feedback on the new example.

Goal: build a function that learns quickly

Online algorithm performance is measured by finding their learning rate. This can be done by measuring accumulated loss as they process a sequence of examples.

Online Ranking Example: a new netflix user with no prior information. Goal: predict new user’s future film demands as s/he expresses views about films watched over time
'Learning to Order Things'
Cohen, Schapire and Singer, NIPS 1998

Definition: Preference function

\[ \text{PREF}(u, v) \in [0, 1] \]

As preference function \( \rightarrow 1 \), this implies \( u \) is strongly preferred over \( v \)

Definition: Rank ordering

\[ R_\rho(u, v) = \begin{cases} 
1 & \text{if } \rho(u) > \rho(v) \\
0 & \text{if } \rho(v) > \rho(u) \\
\frac{1}{2} & \text{otherwise}
\end{cases} \]

\[ \PREF(u, v) \in [0, 1] \]

\[ R_\rho(u, v) = \begin{cases} 
1 & \text{if } \rho(u) > \rho(v) \\
0 & \text{if } \rho(v) > \rho(u) \\
\frac{1}{2} & \text{otherwise}
\end{cases} \]

However, a greedy approach can give a good approximate optimal ordering \( \rho \).
Greedy ordering algorithm

Algorithm Order By Preferences

Inputs: an instance set $X$, a preference function $PREF$
Output: an approximately optimal ordering function $\hat{\rho}$

1. let $V = X$
2. for each $v \in V$ do $\pi(v) = \sum_{u \in V} PREF(v, u) - \sum_{u \in V} PREF(u, v)$
3. while $V$ is non-empty do
   1. let $i = \arg \max_v \pi(v)$ ... find $i$ in $V$ such that $v$ is preferred over $i$ more times than any other member in $V$
   2. let $\hat{\rho}(i) = |V|$ ... then the rank of $i$ is defined by the current size of $V$
   3. $V = V - \{i\}$ ... update $V$ so that $i$ is removed
   4. for each $v \in V$ do $\pi(v) = \pi(v) + PREF(i, v) - PREF(v, i)$
4. endwhile

Figure 1: A greedy ordering algorithm

In words ... you are iteratively removing the worst ranked examples where 'worst' is determined by the PREF function defined on the current set $V$. But assigning ranks and removing the 'worst' first does not necessarily lead to an optimal solution ... hence greedy.

The ranking experts problem

Problem: Given a set of $N$ ranking experts that provide us with a set of orderings $R_i$ for some data, we want to learn the optimal $PREF$ function with weights $w$ assigned for each expert:

$$PREF(u, v) = \sum_i w_i R_i(u, v)$$

Online framework used: At iteration $t$,

INPUTS: $X^t$ instances to be ranked
OUTPUT: $\rho_i$ of $X^t$
FEEDBACK: count of the number of times the ordering of items $u$ and $v$ is agreed, then update weights accordingly
Results – www searches

Two queries to learn:

1. Learning the home pages of ML researchers – 210 searches, 16 experts defined
2. Learning the home pages of Universities – 290 searches, 22 experts defined

Each expert returns the top 30 ranked documents.

Goal is to use the expert rankings to come up with an optimal ranking where the correct webpage is top ranked.

Authors run three experiments:
(i) Use a full feedback learning system (online learning as described before)
(ii) ‘click data’ learning system (done by observing the user’s interactions)
(iii) Choosing the best expert

---

Results

<table>
<thead>
<tr>
<th></th>
<th>ML Domain</th>
<th></th>
<th></th>
<th>University Domain</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Top 1</td>
<td>Top 10</td>
<td>Top 30</td>
<td>Av. rank</td>
<td>Top 1</td>
<td>Top 10</td>
</tr>
<tr>
<td>Learned System (Full Feedback)</td>
<td>114</td>
<td>185</td>
<td>198</td>
<td>4.9</td>
<td>111</td>
<td>225</td>
</tr>
<tr>
<td>Learned System (“Click Data”)</td>
<td>95</td>
<td>185</td>
<td>198</td>
<td>4.9</td>
<td>87</td>
<td>229</td>
</tr>
<tr>
<td>Naive</td>
<td>89</td>
<td>165</td>
<td>176</td>
<td>7.7</td>
<td>59</td>
<td>157</td>
</tr>
<tr>
<td>Best (Top 1)</td>
<td>119</td>
<td>170</td>
<td>184</td>
<td>6.7</td>
<td>112</td>
<td>221</td>
</tr>
<tr>
<td>Best (Top 10)</td>
<td>114</td>
<td>182</td>
<td>190</td>
<td>5.3</td>
<td>111</td>
<td>223</td>
</tr>
<tr>
<td>Best (Top 30)</td>
<td>97</td>
<td>181</td>
<td>194</td>
<td>5.6</td>
<td>111</td>
<td>223</td>
</tr>
<tr>
<td>Best (Av. Rank)</td>
<td>114</td>
<td>182</td>
<td>190</td>
<td>5.5</td>
<td>111</td>
<td>223</td>
</tr>
</tbody>
</table>

Table 1: Comparison of learned systems and individual search queries