Multi-prototype Label Ranking with Novel Pairwise-to-Total-Rank Aggregation

Mihajlo Grbovic*
Yahoo! Labs

Nemanja Djuric*, Slobodan Vucetic
Department of Computer and Information Sciences, Temple University

Abstract

We propose a multi-prototype-based algorithm for online learning of soft pairwise preferences over labels. The algorithm learns soft label preferences via minimization of the proposed soft rank-loss measure, and can learn from total orders as well as from various types of partial orders. The soft pairwise preference algorithm outputs are further aggregated to produce a total label ranking prediction using a novel aggregation algorithm that outperforms existing aggregation solutions. Experiments on synthetic and real-world data demonstrate the state-of-the-art performance of the proposed model.

What is Label Ranking?

In Label Ranking, the input space is defined by a feature vector **x** and the output is defined as a ranking $\boldsymbol{\pi}$ of *L* labels (e.g., $\boldsymbol{\pi}$ = [3 1 4 2]).

Given a sample from the underlying distribution **D** = {($\mathbf{x}_n$, $\boldsymbol{\pi}_n$), *n* = 1,…,*N*}, where $\mathbf{x}_n$ is a d-dimensional feature vector and $\boldsymbol{\pi}_n$ is a vector containing either a total or a partial order of a finite set *Y* of *L* class labels, the goal is to learn a model f(x):

$$f(\mathrm{x}) \rightarrow \pi$$

It is related but different from regression and (multi-class) classification, learning to rank and collaborative filtering. Multi-label classification is a special case – label group of preferred over other group (bipartite ranking)

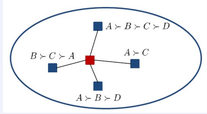
Example 1: users described by internet activity features **x** (website visits, ad clicks, purchases) predict ranking $\boldsymbol{\pi}$ of *L* topics that reflect the user reading preferences

Example 2: customers described by demographic and geo features **x** (age, gender, location) predict ranking $\boldsymbol{\pi}$ of *L* products (e.g., sushis) that reflect the costumer preferences

Related work

1. **Map into classification:**
   - *L* (*L* - 1) / 2 classifiers
   - a single (*d* x *L*) dimensional problem
2. ***k*-NN based algorithms**
   Aggregate neighboring ranks using Plackett-Luce or Mallows
3. **Utility functions:**
   - Learn scoring functions
   - Rank utility scores

$f_k : \mathrm{x} \rightarrow R,\ k = 1,\ldots,L$



Preference Representation

Preferences as an order π

$\pi(i)$ the class label at *i*-th position in the order
$\pi^{-1}(j)$ the position of the y_j class label in the order

Label Ranking (π)				
1	4	2	5	3

Disadvantages: How to handle partial rankings, partial preferences that do not form order, indifferences, weighted preferences?

Preferences as a preference matrix Y

y_i \ y_j	1	2	3	4	5
1	0	NaN	0	0.5	0
2	NaN	0	0	1	0
3	1	1	0	1	1
4	0.5	0	0	0	0
5	1	1	0	1	0

(*L* x *L*)

Disadvantages:
- Quadratic with *L*
- Memory and speed may be an issue
- Mapping from **Y** to $\boldsymbol{\pi}$ is not straightforward
- Preferences may not be transitive

Loss Function

Kendall tau distance

Counts the number of discordant label pairs,

$$d_\tau(\pi,\rho) = \left|\{(y_i,y_j) : \pi_n^{-1}(y_i) > \pi_n^{-1}(y_j)\} \mid \rho_n^{-1}(y_j) > \rho_n^{-1}(y_i)\}\right|$$

Given data set $D = \{(\mathrm{x}_n, \pi_n), n = 1\ldots N\}$, we minimize the label ranking loss,

$$loss_{LR} = \frac{1}{N}\sum_{n=1}^{N}\frac{2\cdot d_\tau(\pi_n,\hat{\pi}_n)}{L\cdot(L-1)}$$

Ground truth **Y**

y_i \ y_j	1	2	3	4	5
1	0	0	0	0.5	0
2	1	0	0	1	0
3	1	1	0	1	1
4	0.5	0	0	0	0
5	1	1	0	1	0

Predicted $\hat{\mathbf{Y}}$

y_i \ y_j	1	2	3	4	5
1	0	0.2	0.17	0.55	0.37
2	0.8	0	0.71	0.66	0.66
3	0.83	0.29	0	0.78	0.02
4	0.45	0.33	0.22	0	0
5	0.63	0.33	0.98	1	0

$$loss_{LR} = \frac{1}{N}\sum_{n=1}^{N}\frac{\left\|\mathbf{Y}_n - \hat{\mathbf{Y}}_n\right\|_F^2}{L\cdot(L-1)}$$

$\mathbf{Y}(i,j) > \mathbf{Y}(j,i)$,
y_i preferred over y_j

Learning

Goal: Train a model for $f : \mathrm{x}_n \rightarrow \pi_n$ in two steps

1. Learn $h : \mathrm{x}_n \rightarrow \mathrm{Y}_n$

Multi-prototype model for Pairwise Preferences

$$\min\ loss_{LR} = \frac{1}{N}\sum_{n=1}^{N}\frac{\left\|\mathrm{Y}_n - \hat{\mathrm{Y}}_n\right\|_F^2}{L\cdot(L-1)}$$

2. Learn $g : \mathrm{Y}_n \rightarrow \pi_n$

Preference matrix aggregation into full rank

maximize

$$AGREE(\pi,\mathrm{Y}) = \sum_{i,j:\pi^{-1}(i)>\pi^{-1}(j)} \mathrm{Y}(i,j)$$

Soft Multi-prototype Label Ranking (SMP-Rank)

Model is completely defined with *K* prototypes {($\mathbf{m}_k$, $\mathbf{Q}_k$), *k* = 1, …, *K*}; $\mathbf{m}_k$ is a *d*-dimensional vector in input space, and $\mathbf{Q}_k$ is the corresponding pairwise preference matrix.

We propose the following mixture model for the posterior probability *P*(**Y**| **x**).

$$P(\mathrm{Y}\mid\mathrm{x}) = \sum_{k=1}^{K} P(k\mid\mathrm{x})P(\mathrm{Y}\mid k)$$

Probability of prototype *k* generating instance **x** is modeled based on standard Gaussian mixture.

$$P(k\mid\mathrm{x}) = \frac{\exp(-\left\|x-m_k\right\|^2/2\sigma_p^2)}{\sum_{u=1}^{K}\exp(-\left\|x-m_u\right\|^2/2\sigma_p^2)}$$

Probability of generating a preference matrix **Y** by prototype *k*, we assume Gaussian error model as follows,

$$P(\mathrm{Y}\mid k) = N(\mathrm{Y}-\mathrm{Q},\sigma_y^2)$$

Learning by minimizing the negative log-likelihood

$$l(\lambda) = -\frac{1}{N\cdot L(L-1)}\sum_{n=1}^{N}\ln\sum_{k=1}^{K}P(k\mid x_n)N(\mathbf{Y}_n-\mathbf{Q}_k,\sigma_y^2)$$

Prediction is made as a linear combination of prototype matrices based on instance distances to prototypes in feature space.

$$\hat{\mathrm{Y}}_n = \sum_{k=1}^{K}P(k\mid x_n)\mathbf{Q}_k$$

Learns from various types of incomplete preferences, and has preferable time complexity *O*(*NKL*), however large *L* leads to large **Q** (of size *L* x *L*).

Solution: Low-rank prototype preference matrix approximation

$\mathbf{Q}_k = \mathrm{U}_k\Sigma_k\mathrm{V}_k$

U_k, V_k are *L* x *m* matrices,
Σ_k is *m* x *m*, *m* << *L*

Work with large $\mathbf{Q}_k$ while storing smaller U_k V_k Σ_k to memory, reduces noise. Use gradient descent to optimize for matrices.

TOUGH aggregation algorithm

For an instance **x**, the SMP-Rank output is represented by a pairwise preference matrix **Y**. The aggregation algorithm computes the total order given these, often conflicting, pairwise preferences. Since pairwise preferences are not transitive in general, inferring total order of labels is not a trivial task. We propose a novel pairwise-to-total-order aggregation algorithm called TOUGH, which maximizes the agreement between the soft preference matrix **Y** and total order $\boldsymbol{\pi}$ defined as follows,

$$AGREE(\pi,\mathbf{Y}) = \sum_{i,j:\pi^{-1}(i)>\pi^{-1}(j)}\mathbf{Y}(i,j)$$

TOUGH can be viewed as:

$$f\left(\begin{array}{|c|c|c|c|c|}\hline 0 & 0.2 & 0.17 & 0.55 & 0.37\\\hline 0.8 & 0 & 0.71 & 0.66 & 0.66\\\hline 0.83 & 0.29 & 0 & 0.78 & 0.02\\\hline 0.45 & 0.33 & 0.22 & 0 & 0\\\hline 0.63 & 0.33 & 0.98 & 1 & 0\\\hline\end{array}\right) = \boldsymbol{\pi}$$

The pseudo-code of the TOUGH algorithm is given below,

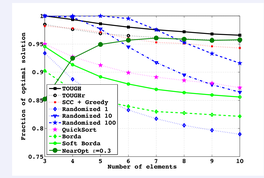
Algorithm 1 Total-Order-Under-Greedy-Heuristic (TOUGH)

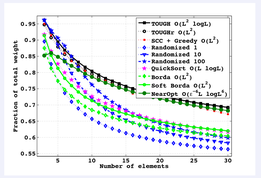
Inputs: label instance set $\Omega = \{1, 2, ..., L\}$, preference matrix **Y**
Output: approximately optimal ordering $\hat{\pi}$

1. **Initialize** $\hat{\pi}$ to empty list
2. **while** (Ω is not empty)
3. **Find** highest value in **Y** and take the preferred label *temp*
4. **Try** every place in $\hat{\pi}$ for *temp*, and put in the place maximizing (9) of $\hat{\pi}$
5. **Set** to 0 entries in **Y** between *temp* and other members of $\hat{\pi}$
6. **Delete** *temp* from Ω

We provide theoretical result that agreement of total order computed by TOUGH algorithm is always within factor of 2 to the agreement of optimal order. Moreover, if in Step 3 we pick random value, theoretical guarantees remain the same with significant speed gains (called TOUGHr algorithm).

TOUGH – Rank-aggregation results





We compared 8 different aggregation algorithms: SCC+Greedy, Borda count, Soft Borda, QuickSort, NearOpt, TOUGH, TOUGHr and Randomized. Randomized algorithm is a heuristic that chooses the better between some random order and its inverse (*L*, 10*L* and 100*L* random orders were compared for each matrix).
We randomly generated 10,000 **Y** matrices of sizes ranging from 3 to 30, where element **Y**(*i*, *j*) is sampled from [0, 1] and **Y**(*i*, *j*) = 1 - **Y**(*i*, *j*). For smaller **Y** we calculated fraction of optimal solution, and for larger matrices we calculated fraction of total weight,

$$\frac{\sum_{i,j:\pi^{-1}(i)>\pi^{-1}(j)}\max\{\mathbf{Y}(i,j)-\mathbf{Y}(j,i),0\}}{\sum_{i,j}\max\{\mathbf{Y}(i,j)-\mathbf{Y}(j,i),0\}}.$$

Label Ranking experimental results

We used 22 data sets; 16 semi-synthetic data sets obtained by converting benchmark multi-class and regression data from the UCI and Statlog repositories into label ranking data using Naive Bayes and feature-to-label technique, and 6 real-world data sets from medical and food domains.

Algorithms compared:
1) SMP-Rank with TOUGH
2) 1-NN rule
3) Instance-based Mallows (IB-Mal)
4) Plackett-Luce (IB-PL)
5) Log-linear models (Lin-LL, Lin-PL)
6) PW (pairwise learning), RBF kernel SVM as base models

Learning from incomplete preferences:
1) 30% and 60% missing labels
2) 30% missing preferences

Results:
SMP-Rank using TOUGH achieved the best overall accuracy and consistently outperformed the competitors. For example, it outperforms the best considered algorithm (PW TOUGH) 16 out of 22 times for 60% missing labels, while the worst-performing algorithm (Lin-LL) is outperformed 20 out of 22 times.

Table 1: Label ranking performance comparison in terms of label rank loss $loss_{LR}$ (**m. p.** - missing preferences)

	complete ranking						30% missing labels						60% missing labels						30% m. p.	
Data Set	**IB** PL	**IB** Mal	**Lin** LL	**Lin** PL	**PW** TGH	**SMP** TGH	**IB** PL	**IB** Mal	**Lin** LL	**Lin** PL	**PW** TGH	**SMP** TGH	**IB** PL	**IB** Mal	**Lin** LL	**Lin** PL	**PW** TGH	**SMP** TGH	**PW** TGH	**SMP** TGH
fried	.202	.196	.344	.347	.180	**.144**	.216	.219	.345	.348	.189	**.171**	.257	.312	.349	.350	.192	**.180**	.188	**.175**
calhous.	.337	**.336**	.449	.435	.348	.346	**.345**	.348	.450	.436	.350	.347	.376	.415	.453	.437	.366	**.349**	.349	**.345**
elevators	.234	.226	.306	.305	**.186**	**.186**	.238	.244	.309	.306	.221	**.193**	.263	.322	.311	.307	.257	**.204**	.219	**.194**
pendigits	.143	.139	.364	.351	**.134**	.152	.152	.159	.366	.354	**.143**	.166	.169	.230	.368	.355	**.153**	.172	**.142**	.165
cpu	.344	**.336**	.378	.361	.337	**.336**	.351	.350	.379	.369	.348	**.337**	.371	.408	.380	.370	.373	**.342**	.345	**.337**
segment	.089	.089	.241	.227	.094	**.070**	.111	.117	.242	.228	.099	**.084**	.124	.154	.242	.226	.121	**.093**	.099	**.097**
stock	.106	.108	.283	.268	.163	**.091**	.134	.133	.284	.269	.163	**.113**	.153	.169	.285	.272	.164	**.142**	.171	**.122**
vehicle	.082	.079	.126	.115	.067	**.064**	.093	.095	.129	.118	.077	**.071**	.113	.131	.128	.125	.091	**.082**	.076	**.071**
authorsh.	.063	.062	.182	.175	.082	**.061**	.068	.072	.189	.178	.086	**.065**	.085	.108	.200	.190	.092	**.085**	.089	**.058**
vowel	.158	.148	.313	.317	.140	**.112**	.217	.178	.316	.318	.167	**.151**	.234	.240	.329	.327	.225	**.191**	.179	**.168**
housing	.329	.334	.398	.385	.334	**.305**	.355	.357	.404	.386	.345	**.328**	.384	.385	.408	.391	.366	**.350**	.343	**.335**
bodyfat	.442	.438	.422	.404	.417	**.409**	.463	.448	.430	**.406**	.419	.436	.466	.460	.431	.429	**.426**	.440	.423	**.418**
glass	.098	.094	.107	.106	.080	**.076**	.107	.111	.111	.108	**.085**	**.085**	.115	.140	.124	.118	.101	**.098**	.107	**.100**
wisconsin	.435	.419	.403	**.401**	.418	.408	.434	.427	.409	**.403**	.438	.431	.445	.443	.422	**.412**	.441	.444	.433	**.400**
wine	.040	.036	.067	.059	.038	**.034**	.041	.042	.071	.064	.046	**.040**	.061	.077	.101	.079	.050	**.041**	.046	**.042**
iris	.040	.037	.189	.171	.089	**.029**	.059	.066	.199	.172	.100	**.037**	.103	.118	.208	.174	.105	**.080**	.111	**.048**
sushi	.348	.349	.459	.462	.335	**.332**	.363	.352	.461	.465	.344	**.335**	.381	.374	.463	.468	.366	**.342**	.339	**.333**
cold	.386	.387	.409	.399	**.370**	.384	.398	.398	.410	.404	**.373**	.380	.424	.423	.411	**.405**	.406	.405	.372	**.369**
diau	.336	.332	.350	.349	**.328**	.337	.338	.340	.353	.350	**.331**	.338	.339	.365	.360	.352	**.335**	.353	.322	**.320**
dtt	.418	.417	.421	.411	**.399**	.409	.428	.430	.422	.417	**.404**	.421	.447	.442	.423	.419	.411	.422	**.404**	.407
heat	.438	.435	.439	.438	.435	**.432**	.445	**.433**	.441	.442	.434	**.433**	.460	.455	.445	.443	**.439**	.452	.426	**.419**
spo	.438	.433	.437	.443	.431	**.430**	.439	.444	.439	.445	.435	**.433**	.441	.455	.444	.453	.441	**.440**	.424	**.419**
average	.250	.247	.322	.315	.246	**.234**	.263	.262	.325	.318	.255	**.245**	.282	.301	.331	.323	.270	**.258**	.256	**.243**

POSTER TEMPLATE BY:
www.PosterPresentations.com