Preference Learning Using ANN

Dariush Hasanpoor

Isfahan University Of Technology

Dec. 2015

Outline

What is preference?

Definition

Preference Learning refers to the task of learning to predict an order relation on a collection of objects (alternatives).

- Preference information plays a key role in automated decision making and appears in various guises in AI researches:
    - Qualitative decision theory
    - Non-monotonic reasoning
    - Constraint satisfaction
    - Planning

Notations

Definition: *Weak Preference*

A weak *preference* relation $\succeq$ on a set $\mathcal{A}$ is a reflexive and transitive binary relation.

Definition: *Strict Preference*

$$a \succ b \longleftrightarrow (a \succeq b) \wedge (b \not\succeq a)$$

- In agreement with preference semantics

Notation	*Interpretation*
$a \succeq b$	"alternative a is at least as preferred as alternative b."
$a \succ b$	"alternative a is preferred over alternative b."

Types of Ranking

- The tasks are categorized as three main problems:
  - **Label ranking**
  - **Object ranking**
  - **Instance ranking**

Types of Ranking

Object Ranking

Task

The task of this model is to find a preference ranking order among instances.

Given

- A (potentially infinite) set X of objects (each object typically represented by a feature vector).
- A finite set of pairwise preferences $x_i \succ x_j$, $(x_i, x_j) \in \mathcal{X} \times \mathcal{X}$.

Find

- A ranking function that, given a set of objects $O \subset X$ as input, returns a permutation(ranking) of these objects.

- In the training phase, preference learning algorithms have access to examples for which the order relation is **(partially)** known.

Preference Learning With The CmpNN

- The method presented in this paper is a pairwise preference learning approach.
- The preference function is implemented by a multilayered feed-forward neural network.
- A CmpNN has following configurations:
  - An input layer with $2d$ units($x,\ y \in \mathbb{R}^d$).
  - One hidden layer.
  - An output layer with two units(the evidence of the relationships $x \succ y$ and $y \succ x$).

Preference Learning With The CmpNN

- Thus, the input to the neural network is the concatenation of the two representations:

$$[x, y] = [x_1, \ldots, x_d, y_1, \ldots, y_d]$$

- The outputs will be denoted by $N_{\succ}([x, y])$ and $N_{\prec}([x, y])$, respectively, where:
    - $N_{\succ}([x, y])$ estimates the evidence of $x \succ y$.
    - $N_{\prec}([x, y])$ estimates the evidence of $y \succ x$.
- The neural network can be trained with the standard back-propagation algorithm.

Preference Learning With The CmpNN

- For each pair of inputs $[x, y]$, the assigned target is:

$$t = [t_1,\ t_2] = \begin{cases} \begin{bmatrix} 1 & 0 \end{bmatrix}^T & x \succ y \\ \begin{bmatrix} 0 & 1 \end{bmatrix}^T & y \succ x \end{cases} \tag{1}$$

- The error is measured by the squared error function:

$$E([x,\ y],\ t) = (t_1 - N_{\succ}([x, y]))^2 + (t_2 - N_{\prec}([x, y]))^2 \tag{2}$$

- After training, the model can be used to predict the preference relationship for an input pair of objects $x,\ y$ as:

$$\begin{cases} x \succ y & N_{\succ}([x, y]) \geq N_{\prec}([x, y]) \\ y \succ x & N_{\succ}([x, y]) \leq N_{\prec}([x, y]) \end{cases}$$

Preference Learning With The CmpNN

- The operators implemented by the preference function, realizes a correct total ordering of the objects only if the following properties hold:
  1. *Reflexivity:* both $x \succ x$ and $x \prec x$ hold.
  2. *Equivalence between $\succ$ and $\prec$:* if $x \succ y$ then $y \succ x$ and vice versa.
  3. *Anti-symmetry:* if $x \succ y$ and $y \succ x$ then $x = y$.
  4. *Transitivity:* if $x \succ y$ and $y \succ z$, then $x \succ z$ (similarly for the $\prec$ relation).
- The proposed method:
  - The reflexivity and the equivalence between $\succ$ and $\prec$ are ensured by the particular architecture adopted for the network.
  - The anti-symmetry property fails only if the two outputs are equal, i.e $N_{\succ}([x, y]) = N_{\prec}([x, y])$ and $x \neq y$ holds. but, since the outputs are real numbers, such an event is very unlikely.
  - The transitivity property is generally hard to be guaranteed by a pairwise preference learning approach and the proposed method does not overcome this limitation.

Preference Learning With The CmpNN

- Non-transitive preference functions can still be adopted to sort a set of objects:
  - Classical sorting algorithms compare only a small subset of all the possible pairs and return an ordering consistent with those comparisons. If an algorithm knows, from the comparator response, that $x \succ y$ and $y \succ z$ hold, then $x \succ z$ is usually assumed without a further comparison.

CmpNN Architecture

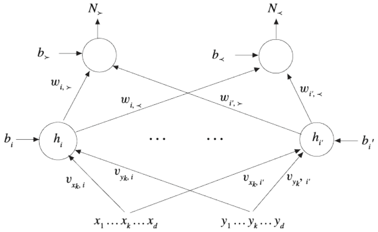

Figure 1 : CmpNN architecture.

- The CmpNN architecture adopts a weight-sharing technique in order to ensure that the reflexivity and the equivalence between $\succ$ and $\prec$ hold.

CmpNN Architecture

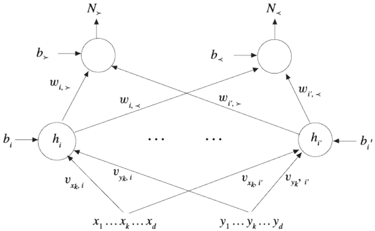

Figure 1 : CmpNN architecture.

- Assuming that:
  - $v_{x_k,i}(v_{y_k,i})$ denotes the weight of the connection from the input node $x_k(y_k)$ to ith hidden node.
  - $w_{i,\succ}(w_{i,\prec})$ represent the weights of the connections from the ith hidden to the output nodes.

CmpNN Architecture

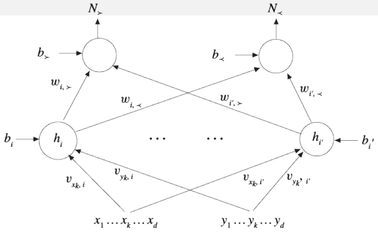

Figure 1 : CmpNN architecture.

- For each hidden neuron i, a dual neuron i' exists whose weights are shared with i according to the following schema:

  1. $v_{x_k,i'} = v_{y_k,i}$ and $v_{y_k,i'} = v_{x_k,i}$ hold, i.e., the weights from $x_k,\ y_k$ to i are swapped in the connections to i'.
  2. $w_{i',\succ} = w_{i,\prec}$ and $w_{i',\prec} = w_{i,\succ}$ hold, i.e., the weights of the connections from the hidden i to the outputs $N_\succ,\ N_\prec$ are swapped in the connections leaving from i'.
  3. $b_i = b_{i'}$ and $b_\succ = b_\prec$ hold, i.e., the biases are shared between the dual hiddens i and i' and between the outputs $N_\succ$ and $N_\prec$.

Learning Algorithm

- The CmpNN is embedded into SortNet, as a comparison function.
- Thus, SortNet can order a set of objects in $\mathcal{O}(n \log n)$.

Learning Algorithm

- The CmpNN is trained on a dataset, composed of pairs of objects.
- The comparative neural network is trained using the square error function that forces the network outputs to be close to the desired targets on each single pair of objects.
- In general, the optimization of the square error does not necessarily correspond to a good ranking.
- A good training procedure should use, in some way, a measure of the quality of the global ordering.
- If the number n of the objects is large, then the inclusion of all the $\binom{n}{2}$ pairs into the learning set can make the training very slow or this number of training even may not be useful at all.

Learning Algorithm

Algorithm 1 SortNet learning algorithm

$T \leftarrow \textit{Set of training objects}$
$V \leftarrow \textit{Set of validation objects}$
$P_T^0 \leftarrow \emptyset;$
$P_V^0 \leftarrow \emptyset;$
$C^0 \leftarrow RandomWeightNetwork();$
for $i = 0$ to max_iter **do**
  if $i \geq 1$ **then**
    $C^i \leftarrow TrainAndValidate(P_T^i, P_V^i);$
  end if
  $[E_T^i\ ,\ R_T^i] \leftarrow Sort(C^i, T);$
  $[E_V^i\ ,\ R_V^i] \leftarrow Sort(C^i, V);$
  $score \leftarrow RankQuality(R_V^i);$
  if $score > best\_score$ **then**
    $best\_score \leftarrow score;$
    $C^* \leftarrow C^i;$
  end if
  $P_T^{i+1} \leftarrow P_T^i \cup E_T^i;$
  $P_V^{i+1} \leftarrow P_V^i \cup E_V^i;$
  if $P_T^{i+1} = P_T^i$ and $P_V^{i+1} = P_V^i$ **then**
    return $C^*;$
  end if
end for
return $C^*;$

Measures of the Ranking Quality

- This paper considered the three measures proposed for the *LETOR* benchmark:
  - **Precision at position n (P@n):** This value measures the relevance of the top n results of the ranking list with respect to a given query:

$$\text{P@n} = \frac{\text{relevant docs in top n results}}{n}$$

  - **Mean average precision (MAP):** Given a query q, the average precision is:

$$\text{AP}_q = \frac{\sum_{n=1}^{N_q} \text{P@n} \cdot \text{rel}(n)}{N_q}$$

    rel(n) is 1 if the nth document in the ordering is relevant and 0 otherwise. Thus, AP_q averages the values of P@n over the positions n of the relevant documents.

  - **Normalized discount cumulative gain (NDCG@n):** This measure exploits an explicit rating of the documents in the list.

$$\text{NDCG@n} \equiv Z_n \left((2^{r_1} - 1) + \sum_{j=2}^{n} \frac{2^{r_j} - 1}{\log(j)} \right)$$

    where $r_j \geq 0$ is the rating of the jth document (smaller values indicate less relevance), and Z_n is a normalization factor chosen such that the ideal ordering gets a NDCG@n score of 1.

LATOR Dataset

- A package of benchmarks for LETOR, released by Microsoft Research Asia and available on the Web.
- The task considered in LETOR is that of LETOR, according to their relevance, the documents returned by information retrieval systems in response to a query.
- This paper, considered the version 2.0 of LETOR, which contains three benchmarks:
  - *The TD2003 dataset:* consists of 50 sets of documents, each one containing 1000 documents returned in response to a query (i.e., 50 different queries).
  - *The TD2004 dataset:* contains 75 sets of 1000 documents, each corresponding to a different queries.
  - *The OHSUMED dataset:* is a subset of the medical publication repository MEDLINE consists of 106 sets of documents, with associated relevance degrees, that have been returned in response to queries.

LATOR Dataset

- For TD2003-2004 datasets:
  - Each query-document pair is represented by 44 values that include several features commonly used in information retrieval.
  - A label is assigned to each document to specify whether it is relevant (R) or not (NR) with respect to the given query.
  - For all queries, the relevant documents are roughly 1% of the whole set of the available documents.
- For OHSUMED dataset:
  - The relevance degree is provided by humans and consists in one of the following categories: relevant R, possibly relevant PR, and non-relevant NR.
  - The documents are represented using a set of 25 features.

Experimental Results

TABLE I
COMPARISON OF THE CLASSIFICATION ACCURACY (IN PERCENTAGE) BETWEEN THE CmpNN AND THE MODEL PROPOSED IN [20]

Hidden neurons		CmpNN	Neural Network of [20]
10	Test	**87.46 ± 1.67**	50.75 ± 29.11
	Train	**90.27 ± 1.41**	50.29 ± 29.53
	Validation	**87.09 ± 1.42**	49.56 ± 29.00
20	Test	**88.25 ± 0.56**	60.71 ± 12.15
	Train	**90.73 ± 0.39**	61.43 ± 12.06
	Validation	**88.02 ± 0.38**	60.02 ± 12.22
30	Test	**88.21 ± 0.95**	61.36 ± 19.27
	Train	**90.65 ± 1.78**	61.56 ± 20.24
	Validation	**87.63 ± 1.27**	62.02 ± 18.58
40	Test	**87.95 ± 0.74**	68.14 ± 7.15
	Train	**90.97 ± 1.85**	69.55 ± 7.68
	Validation	**87.93 ± 0.80**	69.14 ± 6.60
50	Test	**88.42 ± 1.08**	68.36 ± 12.01
	Train	**91.40 ± 2.06**	69.64 ± 11.95
	Validation	**88.07 ± 1.36**	68.88 ± 11.90

- The two models were compared on the task of learning a nontransitive preference function over an artificial dataset.
- The results have been averaged using 5 fold cross validation and for each trial we randomly selected 10,000 pairs for training, 4000 pairs for validation, and 6000 pairs for testing.
- The t-student test (with p-value less than 0.05) proves that the model proposed in this paper is able to learn a non-transitive preference function with a significant improvement with respect to the model proposed in [20].

Experimental Results

TABLE III

ACCURACY, PRECISION AND RECALL (IN PERCENTAGE) OF THE PREFERENCE PREDICTION BY THE CmpNN ON THE TEST SET

Hiddens		TD2003	TD2004	OHSUMED
10	Accuracy	**92.93** $\pm$ **3.11**	97.22 $\pm$ 2.25	**94.38** $\pm$ **2.73**
	Precision	**92.95** $\pm$ **2.40**	97.15 $\pm$ 1.50	**94.93** $\pm$ **1.82**
	Recall	**92.95** $\pm$ **2.41**	97.15 $\pm$ 1.50	**94.93** $\pm$ **1.82**
20	Accuracy	90.66 $\pm$ 3.80	**97.97** $\pm$ **1.98**	94.28 $\pm$ 1.74
	Precision	90.11 $\pm$ 2.20	**97.64** $\pm$ **1.32**	94.11 $\pm$ 1.17
	Recall	90.10 $\pm$ 2.20	**97.65** $\pm$ **1.33**	94.19 $\pm$ 1.17
30	Accuracy	88.65 $\pm$ 3.94	96.63 $\pm$ 2.12	93.92 $\pm$ 1.45
	Precision	89.43 $\pm$ 2.29	97.08 $\pm$ 1.41	93.95 $\pm$ 0.96
	Recall	89.43 $\pm$ 2.29	97.08 $\pm$ 1.41	93.95 $\pm$ 0.97

Experimental Results

TABLE IV
SELECTED NUMBER OF HIDDEN NEURONS FOR THE DIFFERENT DATASETS AND RANKING QUALITY MEASURES

Dataset		Hidden neurons
TD2003	SortNet MAP	20
	SortNet P@10	30
	SortNet NDCG@10	20
TD2004	SortNet MAP	20
	SortNet P@10	10
	SortNet NDCG@10	10
OHSUMED	SortNet MAP	20
	SortNet P@10	10
	SortNet NDCG@10	10

Experimental Results

TABLE V
RESULTS ON TD2003 MEASURED BY $NDCG@n$ AND $P@n$

	NDCG@n										P@n									
	n=1	**n=2**	**n=3**	**n=4**	**n=5**	**n=6**	**n=7**	**n=8**	**n=9**	**n=10**	**n=1**	**n=2**	**n=3**	**n=4**	**n=5**	**n=6**	**n=7**	**n=8**	**n=9**	**n=10**
RankBoost	0.26	0.28	0.27	0.27	0.28	0.28	0.29	0.28	0.28	0.29	0.26	0.27	0.24	0.23	0.22	0.21	0.21	0.19	0.18	0.18
RankSVM	0.42	0.37	0.38	0.36	0.35	0.34	0.34	0.34	0.34	0.34	0.42	0.35	0.34	0.3	0.26	0.24	0.23	0.23	0.22	0.21
Frank-c19.0	0.44	0.39	0.37	0.34	0.33	0.33	0.33	0.33	0.34	0.34	0.44	0.37	0.32	0.26	0.23	0.22	0.21	0.21	0.2	0.19
ListNet	0.46	0.43	0.41	0.39	0.38	0.39	0.38	0.37	0.38	0.37	0.46	0.42	0.36	0.31	0.29	0.28	0.26	0.24	0.23	0.22
AdaRank MAP	0.42	0.32	0.29	0.27	0.24	0.23	0.22	0.21	0.2	0.19	0.42	0.31	0.27	0.23	0.19	0.16	0.14	0.13	0.11	0.1
AdaRank NDCG	0.52	0.41	0.37	0.35	0.33	0.31	0.3	0.29	0.28	0.27	0.52	0.4	0.35	0.31	0.27	0.24	0.21	0.19	0.17	0.16
SortNet MAP	**0.58**	**0.46**	**0.43**	**0.4**	**0.4**	**0.39**	**0.4**	**0.39**	**0.39**	**0.39**	**0.58**	**0.45**	**0.39**	**0.33**	**0.31**	**0.29**	**0.28**	**0.27**	**0.26**	**0.24**
SortNet P@10	**0.44**	**0.42**	**0.38**	**0.37**	**0.37**	**0.37**	**0.37**	**0.37**	**0.36**	**0.36**	**0.44**	**0.4**	**0.33**	**0.3**	**0.29**	**0.27**	**0.26**	**0.25**	**0.23**	**0.22**
SortNet NDCG@10	**0.5**	**0.42**	**0.41**	**0.39**	**0.39**	**0.39**	**0.39**	**0.39**	**0.38**	**0.38**	**0.5**	**0.41**	**0.38**	**0.34**	**0.31**	**0.3**	**0.29**	**0.27**	**0.26**	**0.24**

Experimental Results

TABLE VI

RESULTS ON TD2004 MEASURED BY $NDCG@n$ AND $P@n$

	NDCG@n										P@n									
	n=1	**n=2**	**n=3**	**n=4**	**n=5**	**n=6**	**n=7**	**n=8**	**n=9**	**n=10**	**n=1**	**n=2**	**n=3**	**n=4**	**n=5**	**n=6**	**n=7**	**n=8**	**n=9**	**n=10**
RankBoost	0.48	0.47	0.46	0.44	0.44	0.45	0.46	0.46	0.46	0.47	0.48	0.45	0.4	0.35	0.32	0.3	0.29	0.28	0.26	0.25
RankSVM	0.44	0.43	0.41	0.41	0.39	0.4	0.41	0.41	0.41	0.42	0.44	0.41	0.35	0.33	0.29	0.27	0.26	0.25	0.24	0.23
FRank	0.44	0.47	0.45	0.43	0.44	0.45	0.46	0.45	0.46	0.47	0.44	0.43	0.39	0.34	0.32	0.31	0.3	0.27	0.26	0.26
ListNet	0.44	0.43	0.44	0.42	0.42	0.42	0.43	0.45	0.46	0.46	0.44	0.41	0.4	0.36	0.33	0.31	0.3	0.29	0.28	0.26
AdaRank MAP	0.41	0.39	0.4	0.39	0.39	0.4	0.4	0.4	0.4	0.41	0.41	0.35	0.34	0.3	0.29	0.28	0.26	0.24	0.23	0.22
AdaRank NDCG	0.36	0.36	0.38	0.38	0.38	0.38	0.38	0.38	0.39	0.39	0.36	0.32	0.33	0.3	0.28	0.26	0.24	0.23	0.22	0.21
SortNet MAP	**0.47**	**0.47**	**0.46**	**0.46**	**0.45**	**0.45**	**0.45**	**0.46**	**0.47**	**0.48**	**0.47**	**0.43**	**0.38**	**0.36**	**0.33**	**0.3**	**0.28**	**0.28**	**0.27**	**0.26**
SortNet P@10	**0.39**	**0.43**	**0.42**	**0.44**	**0.44**	**0.45**	**0.46**	**0.46**	**0.46**	**0.47**	**0.39**	**0.4**	**0.36**	**0.36**	**0.33**	**0.32**	**0.31**	**0.28**	**0.27**	**0.26**
SortNet NDCG@10	**0.43**	**0.47**	**0.46**	**0.47**	**0.46**	**0.47**	**0.47**	**0.48**	**0.48**	**0.49**	**0.43**	**0.43**	**0.39**	**0.37**	**0.34**	**0.32**	**0.31**	**0.29**	**0.28**	**0.27**

Experimental Results

TABLE VII
RESULTS ON OHSUMED MEASURED BY $NDCG@n$ AND $P@n$

	NDCG@n										P@n									
	n=1	**n=2**	**n=3**	**n=4**	**n=5**	**n=6**	**n=7**	**n=8**	**n=9**	**n=10**	**n=1**	**n=2**	**n=3**	**n=4**	**n=5**	**n=6**	**n=7**	**n=8**	**n=9**	**n=10**
RankBoost	0.5	0.48	0.47	0.46	0.45	0.44	0.44	0.44	0.43	0.44	0.6	0.6	0.59	0.56	0.54	0.52	0.52	0.5	0.49	0.5
RankSVM	0.5	0.48	0.46	0.46	0.46	0.45	0.45	0.44	0.44	0.44	0.63	0.62	0.59	0.58	0.58	0.56	0.54	0.52	0.52	0.51
Frank-c4.2	0.54	0.51	0.5	0.48	0.47	0.46	0.45	0.45	0.44	0.44	0.67	0.62	0.62	0.58	0.56	0.53	0.51	0.5	0.5	0.49
ListNet	0.52	0.5	0.48	0.47	0.47	0.45	0.45	0.45	0.45	0.45	0.64	0.63	0.6	0.58	0.57	0.54	0.53	0.52	0.51	0.51
AdaRank.MAP	0.54	0.5	0.48	0.47	0.46	0.45	0.44	0.44	0.44	0.44	0.66	0.6	0.58	0.57	0.54	0.53	0.51	0.5	0.5	0.49
AdaRank.NDCG	0.51	0.47	0.46	0.46	0.44	0.44	0.44	0.44	0.44	0.44	0.63	0.6	0.57	0.56	0.53	0.53	0.52	0.51	0.5	0.49
MHR-BC	0.55	0.49	0.49	0.48	0.47	0.46	0.45	0.44	0.44	0.44	0.65	0.61	0.61	0.59	0.57	0.55	0.53	0.51	0.5	0.5
SortNet MAP	**0.52**	**0.48**	**0.47**	**0.46**	**0.45**	**0.45**	**0.44**	**0.44**	**0.44**	**0.44**	**0.63**	**0.58**	**0.57**	**0.56**	**0.55**	**0.55**	**0.53**	**0.52**	**0.51**	**0.5**
SortNet P@10	**0.49**	**0.43**	**0.44**	**0.43**	**0.42**	**0.42**	**0.42**	**0.42**	**0.42**	**0.42**	**0.62**	**0.57**	**0.57**	**0.55**	**0.53**	**0.51**	**0.51**	**0.51**	**0.5**	**0.49**
SortNet NDCG@10	**0.52**	**0.48**	**0.45**	**0.45**	**0.44**	**0.44**	**0.44**	**0.44**	**0.43**	**0.43**	**0.64**	**0.61**	**0.57**	**0.57**	**0.55**	**0.54**	**0.53**	**0.52**	**0.5**	**0.49**

Experimental Results

TABLE VIII

MAP RESULTS ON TD2003, TD2004, AND OHSUMED

	TD2003	TD2004	OSHUMED
RankBoost	0.212	0.384	0.44
RankSVM	0.256	0.35	0.447
Frank-c4.2	0.245	0.381	0.446
ListNet	0.273	0.372	0.45
AdaRank.MAP	0.137	0.331	0.442
AdaRank.NDCG	0.185	0.299	0.442
MHR-BC	NA	NA	0.44
SortNet MAP	**0.307**	**0.391**	**0.442**
SortNet P@10	**0.256**	**0.381**	**0.438**
SortNet NDCG@10	**0.297**	**0.402**	**0.44**

Conclusion

- A new neural architecture for learning a preference function (CmpNN) and an adaptive ranking algorithm (SortNet) have been proposed.
- The SortNet algorithm exploits an iterative procedure aimed at selecting the most informative patterns from the training set.
- The results show that the proposed method compares favorably with other state-of-the-art techniques.

Thank You!

References

L. Rigutini, T. Papini, M. Maggini, and F. Scarselli, "Sortnet: Learning to rank by a neural preference function," *Neural Networks, IEEE Transactions on*, vol. 22, no. 9, pp. 1368–1380, 2011.

J. Furnkranz and E. Hullermeier, *Encyclopedia of Machine Learning*. Springer, 2010, ch. Preference Learning, pp. 789–795.

Johannes Furnkranz and Eyke Hullermeier, "Preference Learning," *Kunstliche Intelligenz*, pp. 60–61, 2005.

E. Hullermeier, J. Furnkranz, W. Cheng, and K. Brinker, *Artificial Intelligence*. ScienceDirect, 2008, ch. Label ranking by learning pairwise preferences.