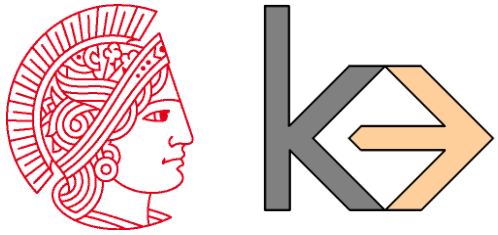
Preference Learning: A Tutorial Introduction

Johannes Fürnkranz
Knowledge Engineering
Dept. of Computer Science
Technical University Darmstadt, Germany

Eyke Hüllermeier
Knowledge Engineering & Bioinformatics Lab
Dept. of Mathematics and Computer Science
Marburg University, Germany



PL-10 @ ECML/PKDD 2010, Barcelona, September 24, 2010

What is Preference Learning ?

- Preference learning is an emerging **subfield of machine learning**
- Roughly speaking, it deals with the **learning of (predictive) preference models** from observed (or extracted) preference information

computer science
artificial intelligence

operations research
social sciences (voting and choice theory)
economics and decision theory

Workshops and Related Events

- NIPS–01: New Methods for Preference Elicitation
- NIPS–02: Beyond Classification and Regression: Learning Rankings, Preferences, Equality Predicates, and Other Structures
- KI–03: Preference Learning: Models, Methods, Applications
- NIPS–04: Learning With Structured Outputs
- NIPS–05: Workshop on Learning to Rank
- IJCAI–05: Advances in Preference Handling
- SIGIR 07–10: Workshop on Learning to Rank for Information Retrieval
- **ECML/PDKK 08–10: Workshop on Preference Learning**
- NIPS–09: Workshop on Advances in Ranking
- American Institute of Mathematics Workshop in Summer 2010: The Mathematics of Ranking

Preferences in Artificial Intelligence

More generally, „preferences“ is a key topic in current AI research

User preferences play a key role in various fields of application:

- recommender systems,
- adaptive user interfaces,
- adaptive retrieval systems,
- autonomous agents (electronic commerce),
- games, …

Preferences in **AI research**:

- **preference representation** (CP nets, GAU networks, logical representations, fuzzy constraints, …)
- **reasoning** with preferences (decision theory, constraint satisfaction, non-monotonic reasoning, …)
- **preference acquisition** (preference elicitation, **preference learning**, ...)

AGENDA

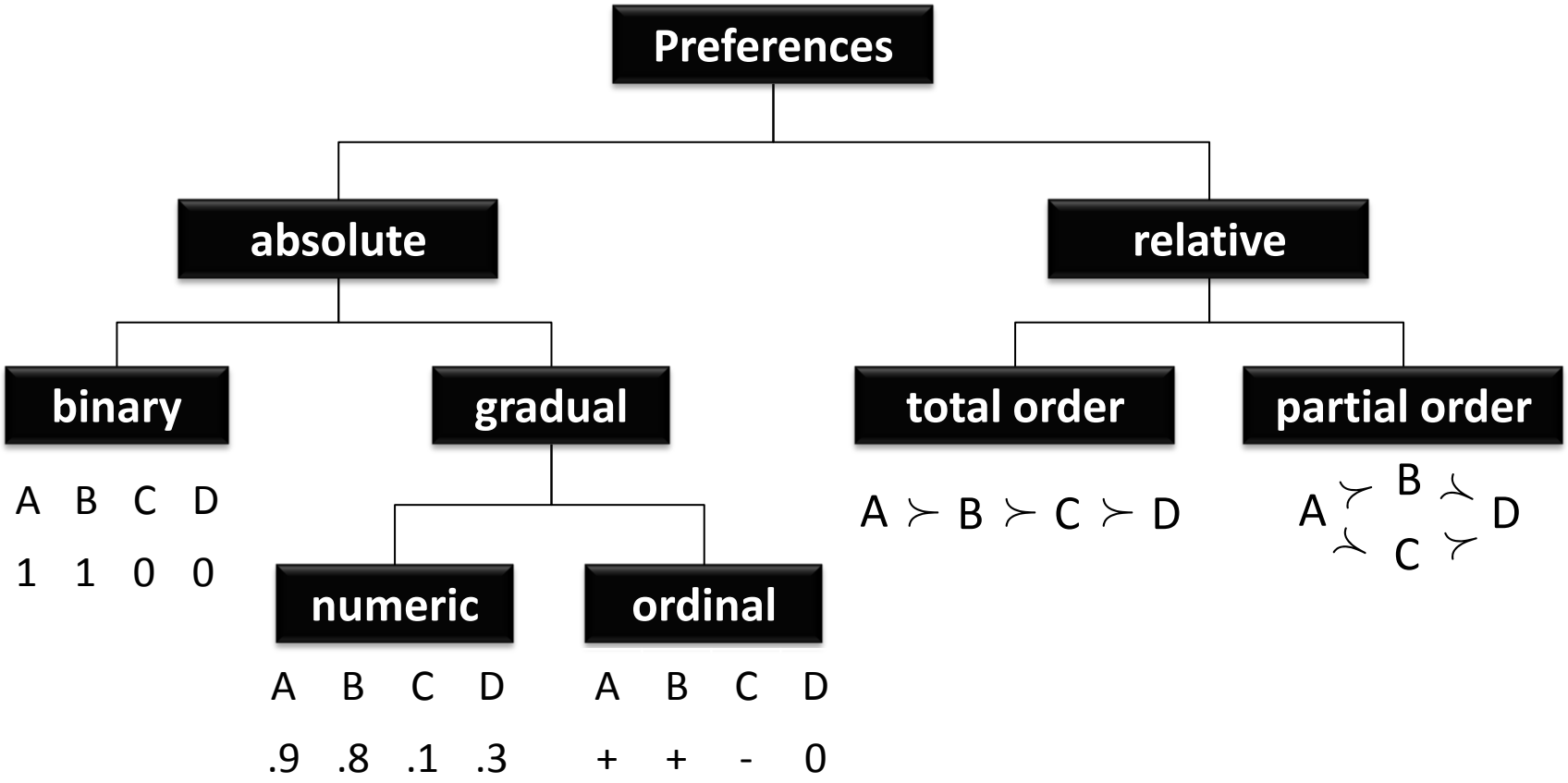
1. Preference Learning Tasks (Eyke)
2. Loss Functions (Johannes)
3. Preference Learning Techniques (Eyke)
4. Complexity of Preference Learning (Johannes)
5. Conclusions

Preference Learning

Preference learning problems can be distinguished along several **problem dimensions**, including

- **representation of preferences, type of preference model**:
  - utility function (ordinal, cardinal),
  - preference relation (partial order, ranking, …),
  - logical representation, …
- **description of individuals/users and alternatives/items**:
  - identifier, feature vector, structured object, …
- **type of training input**:
  - direct or indirect feedback,
  - complete or incomplete relations,
  - utilities, …
- …

Preference Learning



→ (ordinal) regression

→ classification/ranking

Structure of this Overview

(1) Preference Learning as an extension of **conventional supervised learning**: Learn a mapping

$$\mathcal{X} \rightarrow \mathfrak{P}$$

that maps instances to preference models (→ structured/complex output prediction).

(2) Other settings (object ranking, instance ranking, CF, …)

Structure of this Overview

(1) Preference Learning as an extension of **conventional supervised learning**: Learn a mapping

$$\mathcal{X} \rightarrow \mathfrak{P}$$

that maps instances to preference models (→ structured/complex output prediction).

Instances are typically (though not necessarily) characterized in terms of a feature vector.

The output space consists of preference models over a fixed set of alternatives (classes, labels, …) represented in terms of an identifier
→ *extensions of multi-class classification*

Multilabel Classification [Tsoumakas & Katakis 2007]

Training

X1	X2	X3	X4	A	B	C	D
0.34	0	10	174	0	**1**	**1**	0
1.45	0	32	277	0	**1**	0	**1**
1.22	1	46	421	0	0	0	1
0.74	1	25	165	0	**1**	**1**	**1**
0.95	1	72	273	**1**	0	**1**	0
1.04	0	33	158	**1**	**1**	**1**	0

Binary preferences on a fixed set of items: liked or disliked

Prediction

0.92	1	81	382	**0**	**1**	**0**	**1**

LOSS

Ground truth

0.92	1	81	382	**1**	**1**	**0**	**1**

Multilabel Ranking

Training

X1	X2	X3	X4	A	B	C	D
0.34	0	10	174	0	**1**	**1**	0
1.45	0	32	277	0	**1**	0	**1**
1.22	1	46	421	0	0	0	1
0.74	1	25	165	0	**1**	**1**	**1**
0.95	1	72	273	**1**	0	**1**	0
1.04	0	33	158	**1**	**1**	**1**	0

Binary preferences on a fixed set of items: liked or disliked

Prediction

$B \succ D \succ C \succ A$

0.92	**1**	**81**	**382**	**4**	**1**	**3**	**2**

A ranking of all items

LOSS

Ground truth

0.92	**1**	**81**	**382**	**1**	**1**	**0**	**1**

Graded Multilabel Classification [Cheng et al. 2010]

Training

X1	X2	X3	X4	A	B	C	D
0.34	0	10	174	--	+	++	0
1.45	0	32	277	0	++	--	+
1.22	1	46	421	--	--	0	+
0.74	1	25	165	0	+	+	++
0.95	1	72	273	+	0	++	--
1.04	0	33	158	+	+	++	--

Ordinal preferences on a fixed set of items: liked or disliked

Prediction

0.92	**1**	**81**	**382**	**--**	**+**	**0**	**++**

A ranking of all items

LOSS

Ground truth

0.92	**1**	**81**	**382**	**0**	**++**	**--**	**+**

Graded Multilabel Ranking

Training

X1	X2	X3	X4	A	B	C	D
0.34	0	10	174	--	+	++	0
1.45	0	32	277	0	++	--	+
1.22	1	46	421	--	--	0	+
0.74	1	25	165	0	+	+	++
0.95	1	72	273	+	0	++	--
1.04	0	33	158	+	+	++	--

Ordinal preferences on a fixed set of items: liked or disliked

Prediction

$B \succ D \succ C \succ A$

0.92	1	81	382	4	1	3	2

A ranking of all items

LOSS

Ground truth

0.92	1	81	382	0	++	--	+

Label Ranking [Hüllermeier et al. 2008]

Training

X1	X2	X3	X4	Preferences
0.34	0	10	174	$A \succ B, B \succ C, C \succ D$
1.45	0	32	277	$B \succ C$
1.22	1	46	421	$B \succ D, A \succ D, C \succ D, A \succ C$
0.74	1	25	165	$C \succ A, C \succ D, A \succ B$
0.95	1	72	273	$B \succ D, A \succ D,$
1.04	0	33	158	$D \succ A, A \succ B, C \succ B, A \succ C$

Instances are associated with pairwise preferences between labels.

Prediction

$B \succ D \succ C \succ A$

0.92	**1**	**81**	**382**	**4**	**1**	**3**	**2**

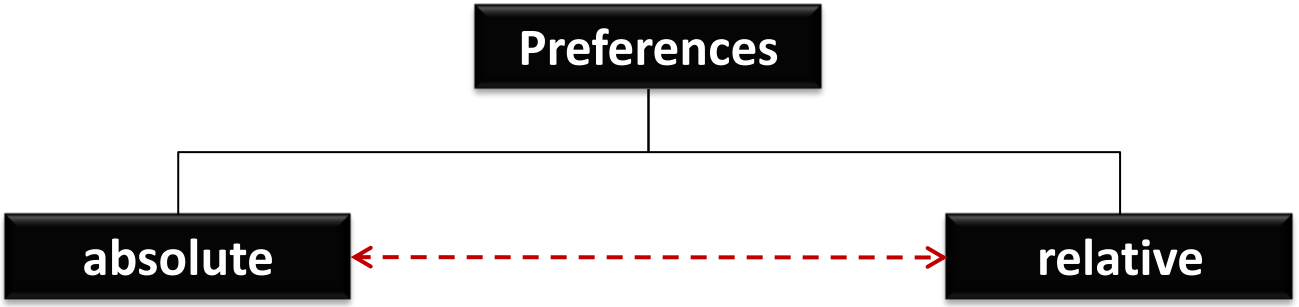
A ranking of all items

LOSS

Ground truth

0.92	**1**	**81**	**382**	**2**	**1**	**3**	**4**

Calibrated Label Ranking [Fürnkranz et al. 2008]



Combining absolute and relative evaluation:

$$a \succ b \succ c \;\mid\; d \succ e \succ f \succ g$$

relevant
positive
liked

irrelevant
negative
disliked

Structure of this Overview

(1) Preference Learning as an extension of conventional supervised learning: Learn a mapping

$$\mathcal{X} \rightarrow \mathfrak{P}$$

that maps instances to preference models (→ structured output prediction).

(2) Other settings

object ranking, instance ranking („no output space“)
collaborative filtering („no input space“)

Object Ranking [Cohen et al. 99]

Training

$$
\begin{array}{lcl}
(0.74, 1, 25, 165) & \succ & (0.45, 0, 35, 155) \\
(0.47, 1, 46, 183) & \succ & (0.57, 1, 61, 177) \\
(0.25, 0, 26, 199) & \succ & (0.73, 0, 46, 185) \\
(0.95, 0, 73, 133) & \succ & (0.25, 1, 35, 153) \\
(0.68, 1, 55, 147) & \succ & (0.67, 0, 63, 182)
\end{array}
$$

Pairwise preferences between objects (instances).

Prediction (ranking a new set of objects)

$$\mathcal{Q} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5, \boldsymbol{x}_6, \boldsymbol{x}_7, \boldsymbol{x}_8, \boldsymbol{x}_9, \boldsymbol{x}_{10}, \boldsymbol{x}_{11}, \boldsymbol{x}_{12}, \boldsymbol{x}_{13}\}$$

$$\boldsymbol{x}_{10} \succ \boldsymbol{x}_4 \succ \boldsymbol{x}_7 \succ \boldsymbol{x}_1 \succ \boldsymbol{x}_{11} \succ \boldsymbol{x}_2 \succ \boldsymbol{x}_8 \succ \boldsymbol{x}_{13} \succ \boldsymbol{x}_9 \succ \boldsymbol{x}_3 \succ \boldsymbol{x}_{12} \succ \boldsymbol{x}_5 \succ \boldsymbol{x}_6$$

Ground truth (ranking or top-ranking or subset of relevant objects)

$$\boldsymbol{x}_{11} \succ \boldsymbol{x}_7 \succ \boldsymbol{x}_4 \succ \boldsymbol{x}_2 \succ \boldsymbol{x}_{10} \succ \boldsymbol{x}_1 \succ \boldsymbol{x}_8 \succ \boldsymbol{x}_{13} \succ \boldsymbol{x}_9 \succ \boldsymbol{x}_{12} \succ \boldsymbol{x}_3 \succ \boldsymbol{x}_5 \succ \boldsymbol{x}_6$$

$$\boldsymbol{x}_{11} \succ \boldsymbol{x}_7 \succ \boldsymbol{x}_4 \succ \boldsymbol{x}_2 \succ \boldsymbol{x}_{10}$$

$$\mathcal{P} = \{\boldsymbol{x}_{11}, \boldsymbol{x}_7, \boldsymbol{x}_4, \boldsymbol{x}_2, \boldsymbol{x}_{10}, \boldsymbol{x}_1\} \qquad \mathcal{N} = \{\boldsymbol{x}_8, \boldsymbol{x}_{13}, \boldsymbol{x}_9, \boldsymbol{x}_{12}, \boldsymbol{x}_3, \boldsymbol{x}_5, \boldsymbol{x}_6\}$$

Instance Ranking [Fürnkranz et al. 2009]

Training

	X1	X2	X3	X4	class
$\boldsymbol{x}_1$	0.34	0	10	174	--
$\boldsymbol{x}_2$	1.45	0	32	277	0
$\boldsymbol{x}_3$	0.74	1	25	165	++
	...	...	...	...	...
$\boldsymbol{x}_n$	0.95	1	72	273	+

Prediction (ranking a new set of objects)

$$\mathcal{Q} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5, \boldsymbol{x}_6, \boldsymbol{x}_7, \boldsymbol{x}_8, \boldsymbol{x}_9, \boldsymbol{x}_{10}, \boldsymbol{x}_{11}, \boldsymbol{x}_{12}, \boldsymbol{x}_{13}\}$$

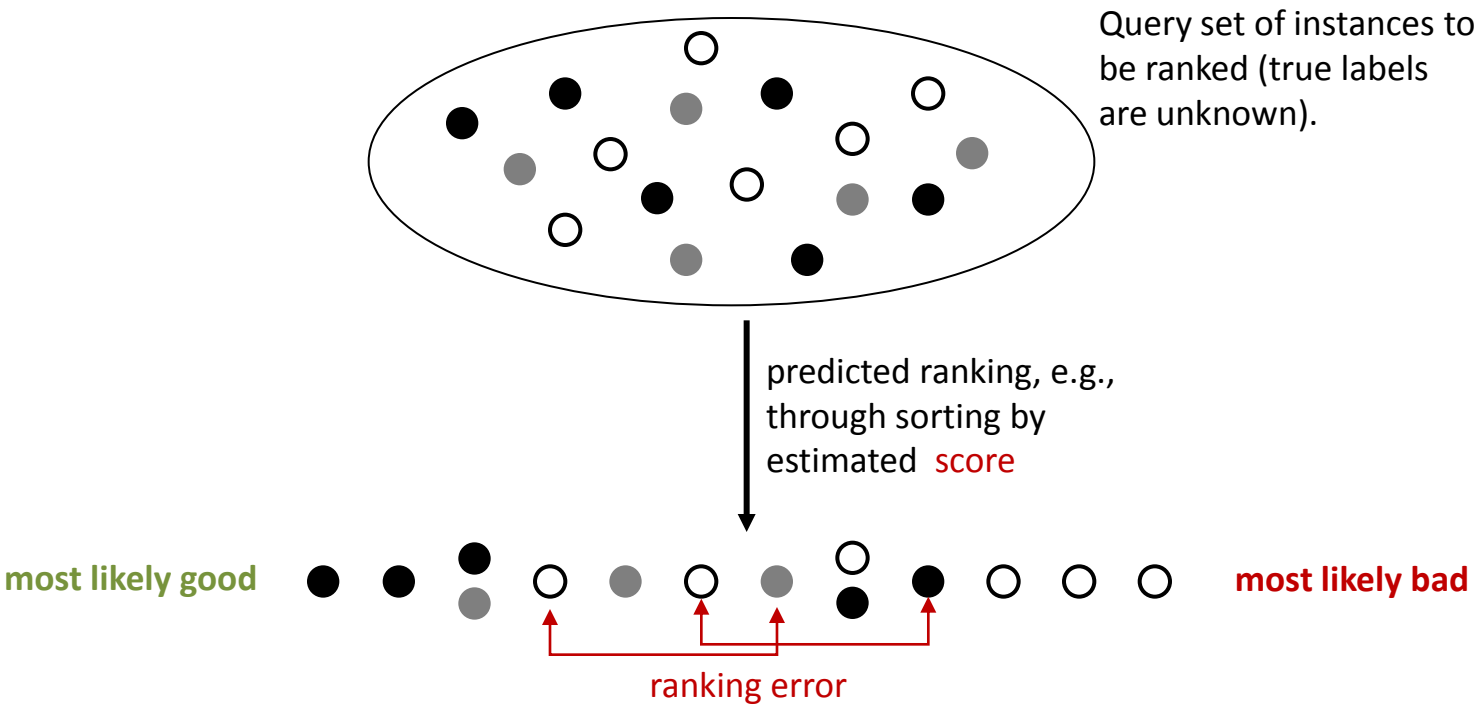
$$\boldsymbol{x}_{10} \succ \boldsymbol{x}_4 \succ \boldsymbol{x}_7 \succ \boldsymbol{x}_1 \succ \boldsymbol{x}_{11} \succ \boldsymbol{x}_2 \succ \boldsymbol{x}_8 \succ \boldsymbol{x}_{13} \succ \boldsymbol{x}_9 \succ \boldsymbol{x}_3 \succ \boldsymbol{x}_{12} \succ \boldsymbol{x}_5 \succ \boldsymbol{x}_6$$

Ground truth (ordinal classes)

$\boldsymbol{x}_{10}$	$\boldsymbol{x}_4$	$\boldsymbol{x}_7$	$\boldsymbol{x}_1$	$\boldsymbol{x}_{11}$	$\boldsymbol{x}_2$	$\boldsymbol{x}_8$	$\boldsymbol{x}_{13}$	$\boldsymbol{x}_9$	$\boldsymbol{x}_3$	$\boldsymbol{x}_{12}$	$\boldsymbol{x}_5$	$\boldsymbol{x}_6$
+	0	++	++	--	+	0	+	--	0	0	--	--

Instance Ranking [Fürnkranz et al. 2009]

Extension of AUC maximization to the polytomous case, in which instances are rated on an ordinal scale such as {**bad, medium, good**}



Collaborative Filtering [Goldberg et al. 1992]

PRODUCTS

USERS	P1	P2	P3	…	P38	…	P88	P89	P90
U1	1		4	…		…		3	
U2		2	2	…		…	1		
…				…		…			
U46	?	2	?	…	?	…	?	?	4
…				…		…			
U98	5			…		…	4		
U99			1	…		…		2	

1: very bad, 2: bad, 3: fair, 4: good, 5: excellent

Inputs and outputs as identifiers, absolute preferences in terms of ordinal degrees.

Preference Learning Tasks

	representation		type of preference information		
task	**input**	**output**	**training**	**prediction**	**ground truth**
collaborative filtering	identifier	**identifier**	absolute ordinal	absolute ordinal	absolute ordinal
multilabel classification	feature	**identifier**	absolute binary	absolute binary	absolute binary
multilabel ranking	feature	**identifier**	absolute binary	ranking	absolute binary
graded multilabel classification	feature	**identifier**	absolute ordinal	absolute ordinal	absolute ordinal
label ranking	feature	**identifier**	relative binary	ranking	ranking
object ranking	**feature**	--	relative binary	ranking	ranking or subset
instance ranking	**feature**	identifier	absolute ordinal	ranking	absolute ordinal

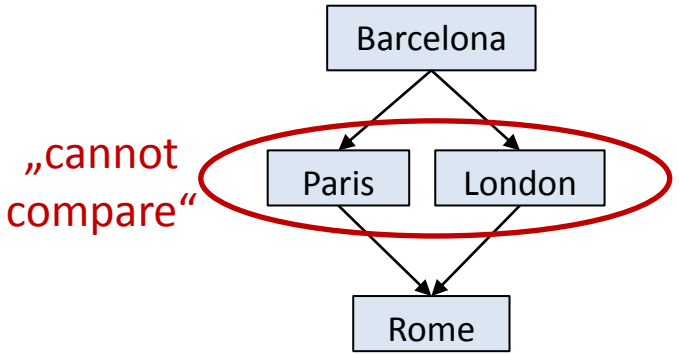
generalized classification

ranking

Two main directions: (1) Ranking and variants (2) generalizations of classification.

Beyond Ranking: Predicting Partial Oders [Chevaleyre et al. 2010, Cheng et al. 2010b]

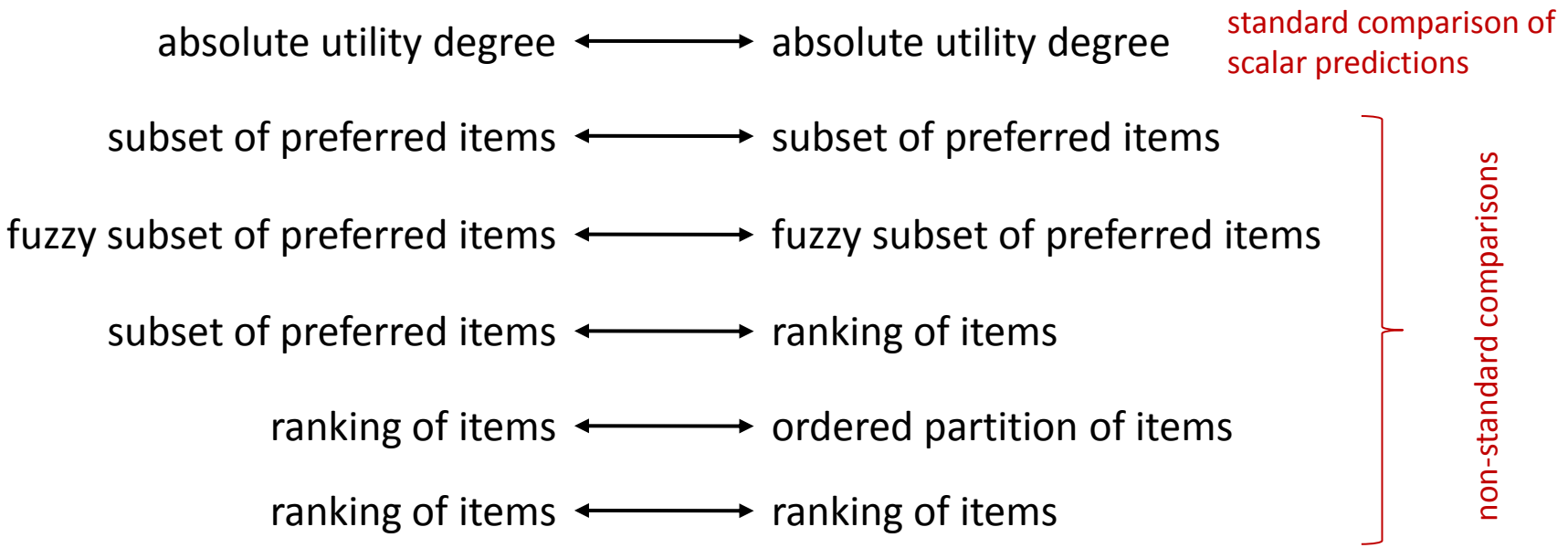
- Rankings (strict total orders) can be generalized in different ways, e.g., through **indifference** (ties) or **incomparability**
- Predicting **partial orders** among alternatives:



- Learning conditional preference (CP) networks
- Two interpretations: **Partial abstention** due to uncertainty (target is a total order) versus prediction of **truly partial order** relation.

Loss Functions

Things to be compared:



References

- W. Cheng, K. Dembczynski and E. Hüllermeier. *Graded Multilabel Classification: The Ordinal Case*. ICML-2010, Haifa, Israel, 2010.
- W. Cheng and E. Hüllermeier. *Predicting partial orders: Ranking with abstention*. ECML/PKDD-2010, Barcelona, 2010.
- Y. Chevaleyre, F. Koriche, J. Lang, J. Mengin, B. Zanuttini. *Learning ordinal preferences on multiattribute domains: The case of CP-nets*. In: J. Fürnkranz and E. Hüllermeier (eds.) Preference Learning, Springer-Verlag, 2010.
- W.W. Cohen, R.E. Schapire and Y. Singer. *Learning to order things*. Journal of Artificial Intelligence Research, 10:243–270, 1999.
- J. Fürnkranz, E. Hüllermeier, E. Mencia, and K. Brinker. *Multilabel Classification via Calibrated Label Ranking*. Machine Learning 73(2):133-153, 2008.
- J. Fürnkranz, E. Hüllermeier and S. Vanderlooy. *Binary decomposition methods for multipartite ranking*. Proc. ECML-2009, Bled, Slovenia, 2009.
- D. Goldberg, D. Nichols, B.M. Oki and D. Terry. *Using collaborative filtering to weave and information tapestry*. Communications of the ACM, 35(12):61–70, 1992.
- E. Hüllermeier, J. Fürnkranz, W. Cheng and K. Brinker. *Label ranking by learning pairwise preferences*. Artificial Intelligence, 172:1897–1916, 2008.
- G. Tsoumakas and I. Katakis. *Multi label classification: An overview*. Int. J. Data Warehouse and Mining, 3:1–13, 2007.

AGENDA

1. Preference Learning Tasks (Eyke)
2. **Loss Functions** (Johannes)
   a. **Evaluation of Rankings**
   b. Weighted Measures
   c. Evaluation of Bipartite Rankings
   d. Evaluation of Partial Rankings
3. Preference Learning Techniques (Eyke)
4. Complexity (Johannes)
5. Conclusions

Rank Evaluation Measures

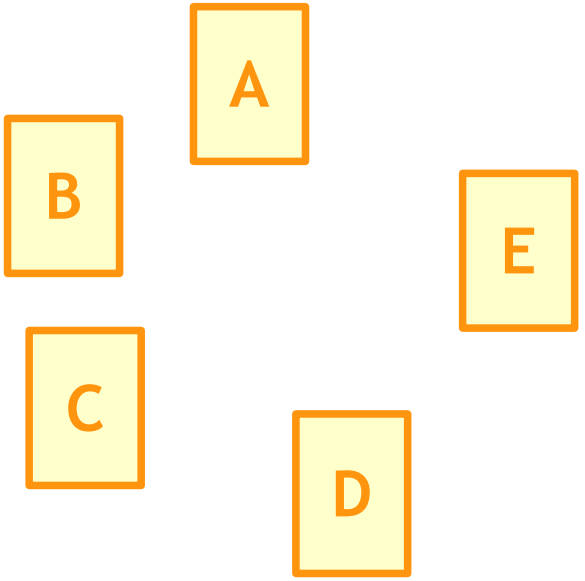
- In the following, we do not discriminate between different ranking scenarios
  - we use the term items for both, objects and labels
- All measures are applicable to both scenarii
  - sometimes have different names according to context
- Label Ranking
  - measure is applied to the ranking of the labels of each examples
  - averaged over all examples
- Object Ranking
  - measure is applied to the ranking of a set of objects
  - we may need to average over different sets of objects which have disjoint preference graphs
    - e.g. different sets of query / answer set pairs in information retrieval

Ranking Errors

- Given:
  - a set of items $X = \{x_1, \ldots, x_c\}$ to rank
    - Example:

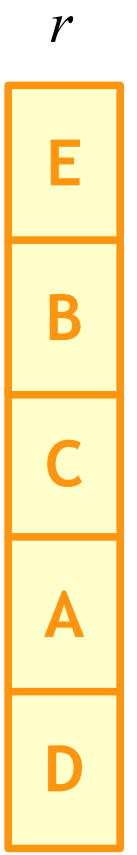
      $X = \{A, B, C, D, E\}$

items can be objects or labels



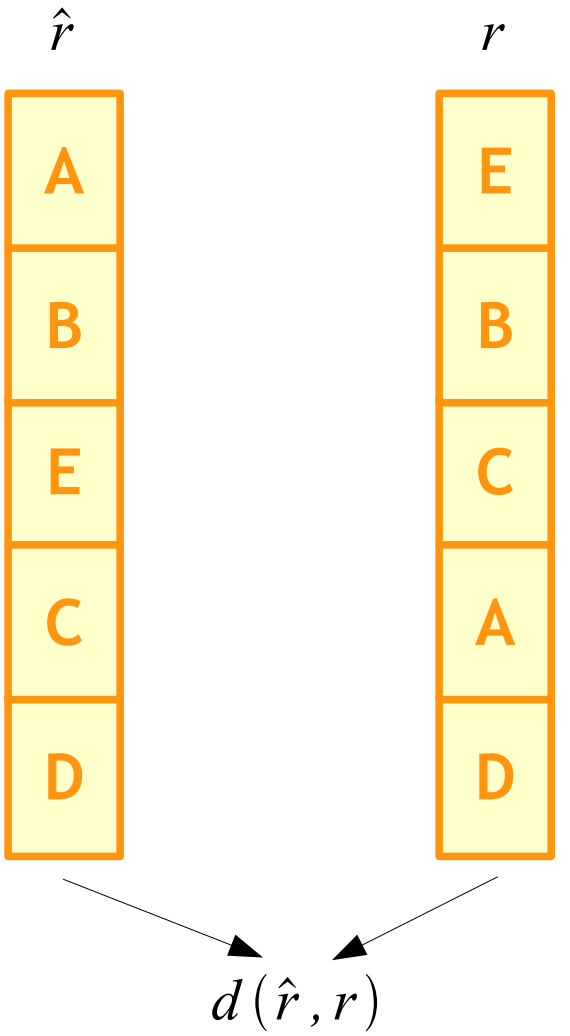
Ranking Errors

- Given:
  - a set of items $X = \{x_1, \ldots, x_c\}$ to rank
    - *Example:*
      X = {A, B, C, D, E}
  - a target ranking r
    - *Example*:
      E ≻ B ≻ C ≻ A ≻ D



Ranking Errors

- Given:
  - a set of items $X = \{x_1, \ldots, x_c\}$ to rank
    - *Example:*
      X = {A, B, C, D, E}
  - a target ranking r
    - *Example*:
      E ≻ B ≻ C ≻ A ≻ D
  - a predicted ranking $\hat{r}$
    - *Example*:
      A ≻ B ≻ E ≻ C ≻ D
- Compute:
  - a value $d(r, \hat{r})$ that measures the *distance* between the two rankings



Notation

- r and $\hat{r}$ are functions from $X \rightarrow \mathbb{N}$
  - returning the rank of an item x

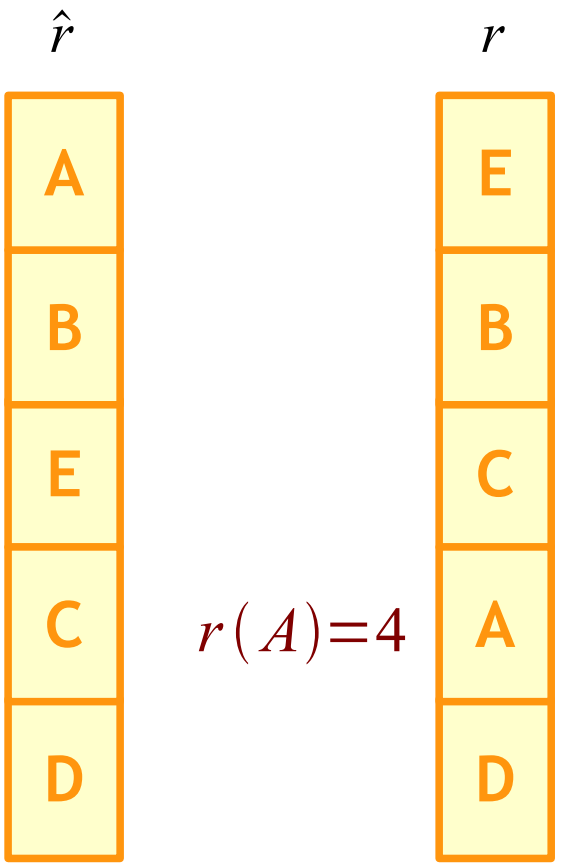
$$\hat{r}(A)=1$$

- the inverse functions r^{-1}: $\mathbb{N} \rightarrow X$
  - return the item at a certain position

$$\hat{r}^{-1}(1)=A \qquad r^{-1}(4)=A$$

- as a short-hand for $r \circ \hat{r}^{-1}$, we also define function R: $\mathbb{N} \rightarrow \mathbb{N}$
  - $R(i)$ returns the true rank of the i-th item in the predicted ranking

$$R(1)=r(\hat{r}^{-1}(1))=4$$

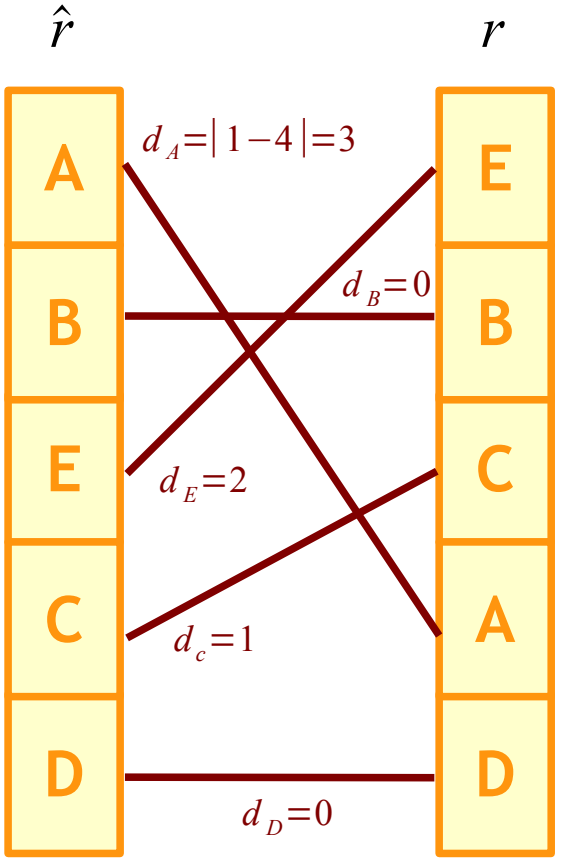


Spearman's Footrule

- Key idea:
  - Measure the sum of absolute differences between ranks

$$D_{SF}(r,\hat{r}) = \sum_{i=1}^{c} |r(x_i) - \hat{r}(x_i)| = \sum_{i=1}^{c} |i - R(i)|$$

$$= \sum_{i=1}^{c} d_{x_i}(r,\hat{r})$$



$$\sum_{x_i} d_{x_i} = 3+0+1+0+2 = 6$$

Spearman Distance

- Key idea:
  - Measure the sum of ~~absolute~~ squared differences between ranks

$$D_S(r,\hat{r}) = \sum_{i=1}^{c} (r(x_i) - \hat{r}(x_i))^2 = \sum_{i=1}^{c} (i - R(i))^2$$

$$= \sum_{i=1}^{c} d_{x_i}(r,\hat{r})^2$$

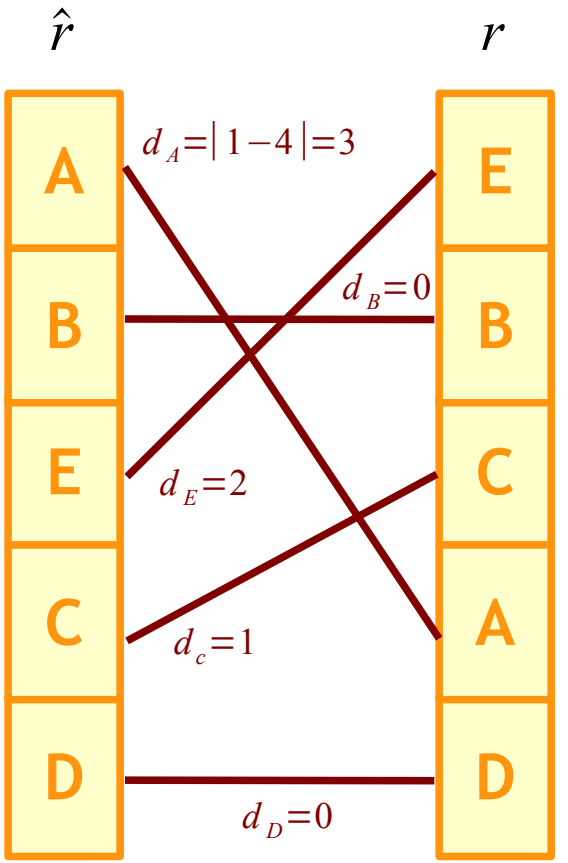
- Value range:

$$\min D_S(r,\hat{r}) = 0$$

$$\max D_S(r,\hat{r}) = \sum_{i=1}^{c} ((c-i)-i)^2 = \frac{c\cdot(c^2-1)}{3}$$

→ Spearman Rank Correlation Coefficient

$$1 - \frac{6\cdot D_S(r,\hat{r})}{c\cdot(c^2-1)} \in [-1,+1]$$

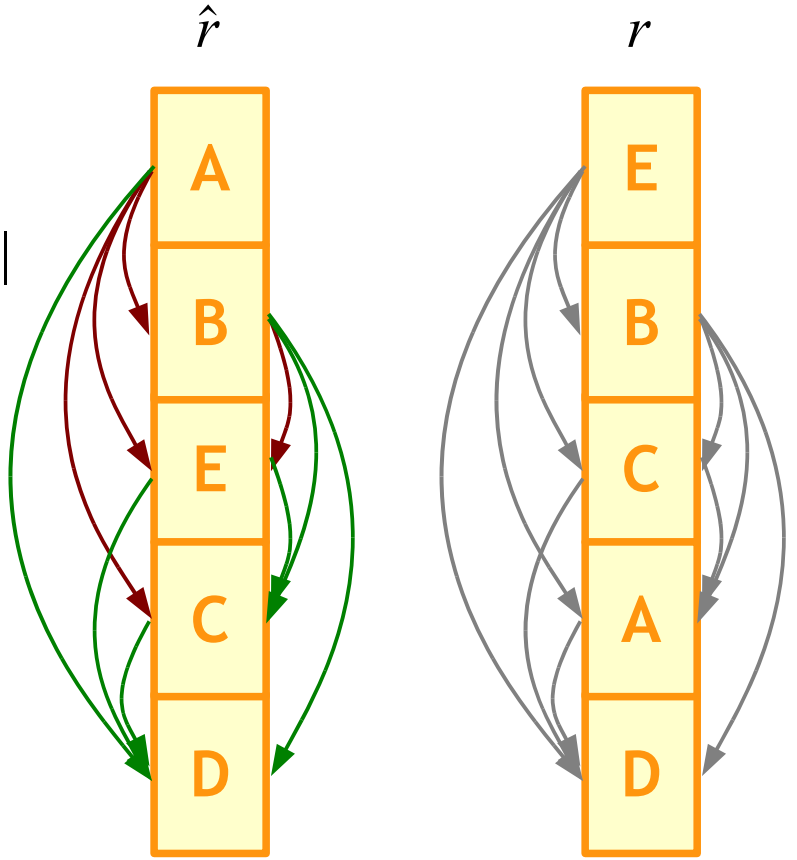


$$\sum_{x_i} d_{x_i}^2 = 3^2 + 0 + 1^2 + 0 + 2^2 = 14$$

Kendall's Distance

- Key idea:
  - number of item pairs that are inverted in the predicted ranking

$$D_\tau(r,\hat{r}) = \left|\left\{(i,j) \mid r(x_i) < r(x_j) \wedge \hat{r}(x_i) > \hat{r}(x_j)\right\}\right|$$



$$D_\tau(r,\hat{r}) = 4$$

- Value range:

$$\min D_\tau(r,\hat{r}) = 0$$

$$\max D_\tau(r,\hat{r}) = \frac{c\cdot(c-1)}{2}$$

→ Kendall's tau

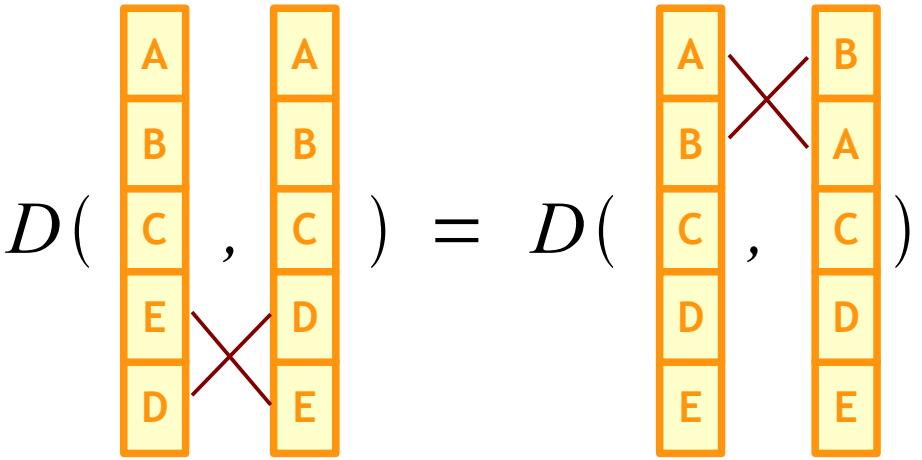
$$1 - \frac{4\cdot D_\tau(r,\hat{r})}{c\cdot(c-1)} \in [-1,+1]$$

AGENDA

1. Preference Learning Tasks (Eyke)
2. **Loss Functions** (Johannes)
    a. Evaluation of Rankings
    b. Weighted Measures
    c. Evaluation of Bipartite Rankings
    d. Evaluation of Partial Rankings
3. Preference Learning Techniques (Eyke)
4. Complexity (Johannes)
5. Conclusions

Weighted Ranking Errors

- The previous ranking functions give equal weight to all ranking positions
  - i.e., differences in the first ranking positions have the same effect as differences in the last ranking positions



- In many applications this is not desirable
  - ranking of search results
  - ranking of product recommendations
  - ranking of labels for classification
  - ...

⇒ Higher ranking positions should be given more weight

Position Error

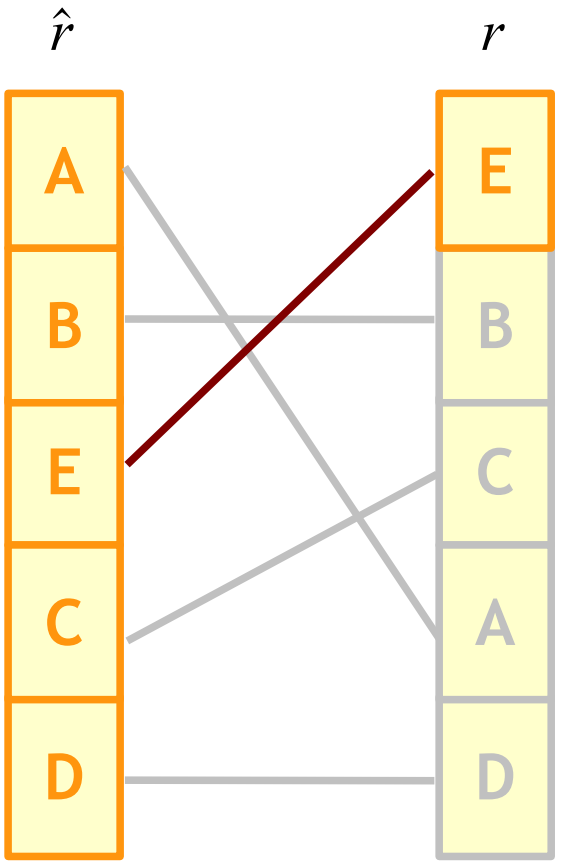
- Key idea:
  - in many applications we are interested in providing a ranking where the target item appears a high as possible in the predicted ranking
    - e.g. ranking a set of actions for the next step in a plan
  - Error is the number of wrong items that are predicted before the target item

$$D_{PE}(r, \hat{r}) = \hat{r}(\arg\min_{x \in X} r(x)) - 1$$

- Note:
  - equivalent to Spearman's footrule with all non-target weights set to 0

$$D_{PE}(r, \hat{r}) = \sum_{i=1}^{c} w_i \cdot d_{x_i}(r, \hat{r})$$

with $w_i = [\![x_i = \arg\min_{x \in X} r(x)]\!]$



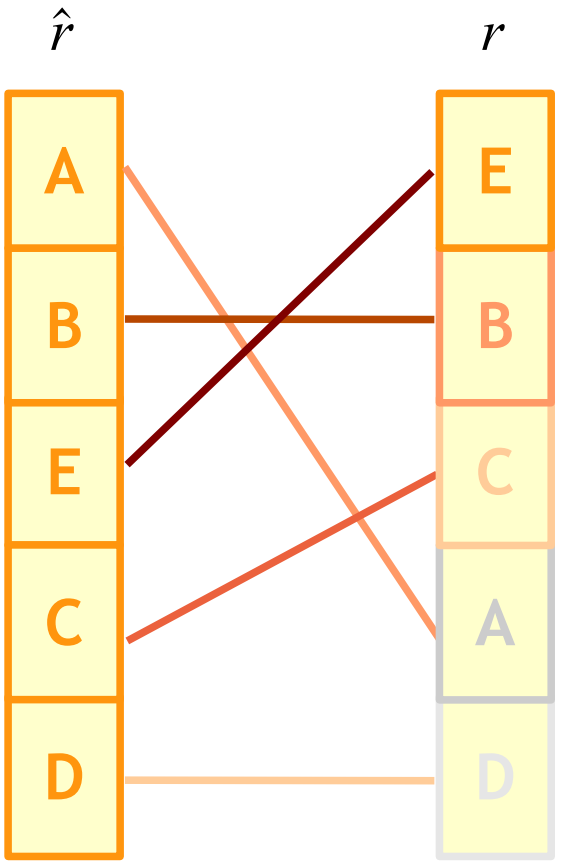
$D_{PE}(r, \hat{r}) = 2$

Discounted Error

- Higher ranks in the target position get a higher weight than lower ranks

$$D_{DR}(r,\hat{r}) = \sum_{i=1}^{c} w_i \cdot d_{x_i}(r,\hat{r})$$

with $w_i = \dfrac{1}{\log(r(x_i)+1)}$



$$D_{DR}(r,\hat{r}) = \frac{3}{\log 2} + 0 + \frac{1}{\log 4} + 0 + \frac{2}{\log 6}$$

(Normalized) Discounted Cumulative Gain

- a "positive" version of discounted error:
  Discounted Cumulative Gain (DCG)

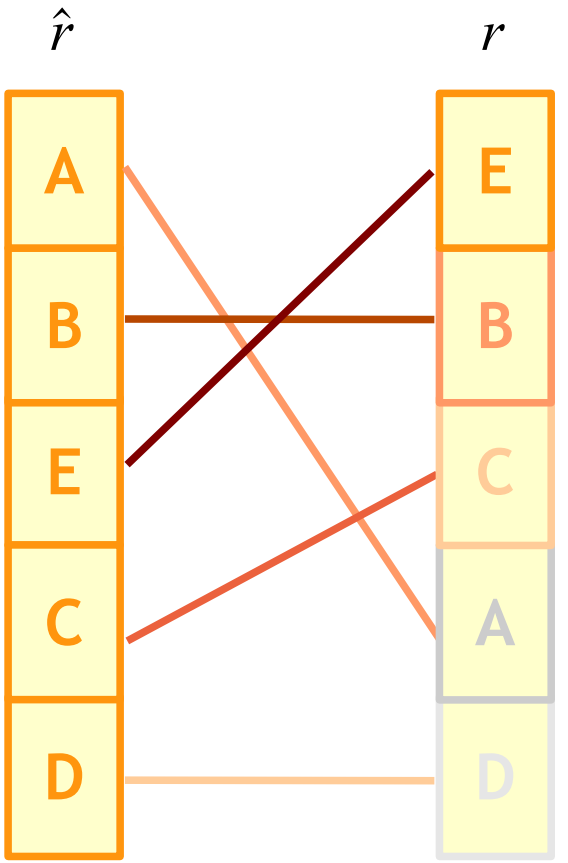
$$DCG(r,\hat{r}) = \sum_{i=1}^{c} \frac{c - R(i)}{\log(i+1)}$$

- Maximum possible value:
  - the predicted ranking is correct,
    i.e. $\forall i : i = R(i)$
  - Ideal Discounted Cumulative Gain (IDCG)

$$IDCG = \sum_{i=1}^{c} \frac{c - i}{\log(i+1)}$$

- Normalized DCG (NDCG)

$$NDCG(r,\hat{r}) = \frac{DCG(r,\hat{r})}{IDCG}$$



$$NDCG(r,\hat{r}) = \frac{\frac{1}{\log 2} + \frac{3}{\log 3} + \frac{4}{\log 4} + \frac{2}{\log 5} + \frac{0}{\log 6}}{\frac{4}{\log 2} + \frac{3}{\log 3} + \frac{2}{\log 4} + \frac{1}{\log 5} + \frac{0}{\log 6}}$$

AGENDA

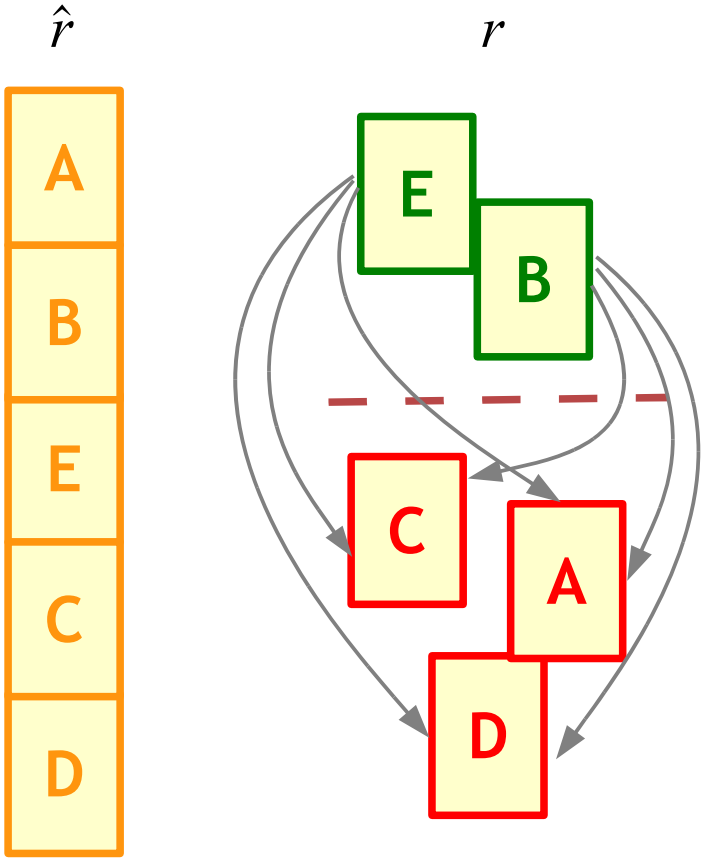
1. Preference Learning Tasks (Eyke)
2. **Loss Functions** (Johannes)
    a. Evaluation of Rankings
    b. Weighted Measures
    c. Evaluation of Bipartite Rankings
    d. Evaluation of Partial Rankings
3. Preference Learning Techniques (Eyke)
4. Complexity (Johannes)
5. Conclusions

Bipartite Rankings

Bipartite Rankings

- The target ranking is not totally ordered but a *bipartite graph*
- The two partitions may be viewed as preference levels $L = \{0, 1\}$
  - all c_1 items of level 1 are preferred over all c_0 items of level 0

- We now have fewer preferences
  - for a total order: $\frac{c}{2} \cdot (c-1)$
  - for a bipartite graph: $c_1 \cdot (c - c_1)$



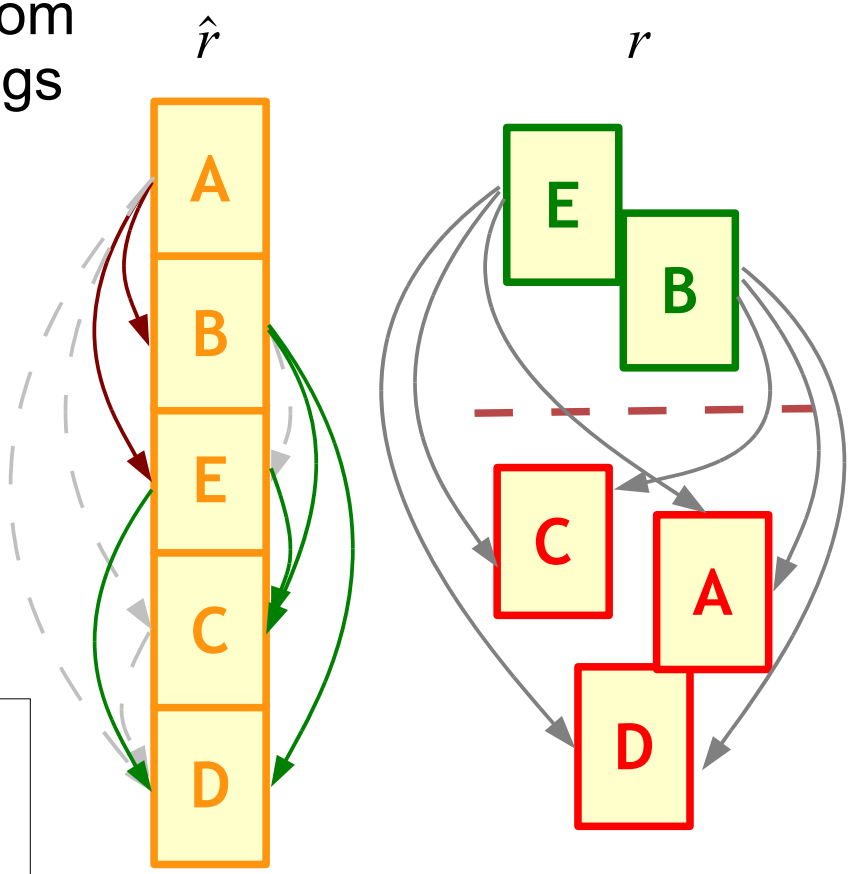
Evaluating Partial Target Rankings

- Many Measures can be directly adapted from total target rankings to partial target rankings
- Recall: Kendall's distance
  - number of item pairs that are inverted in the target ranking

$$D_\tau(r,\hat{r}) = \left| \left\{ (i,j) \mid r(x_i) < r(x_j) \wedge \hat{r}(x_i) > \hat{r}(x_j) \right\} \right|$$

  - can be directly used
  - in case of normalization, we have to consider that fewer items satisfy $r(x_i) < r(x_j)$

- Area under the ROC curve (AUC)
  - the AUC is the fraction of pairs of (p,n) for which the predicted score $s(p) > s(n)$
    - Mann Whithney statistic is the absolute number
  - This is 1 - normalized Kendall's distance for a bipartite preference graph with $L = \{p,n\}$



$$D_\tau(r,\hat{r}) = 2$$

$$AUC(r,\hat{r}) = \frac{4}{6}$$

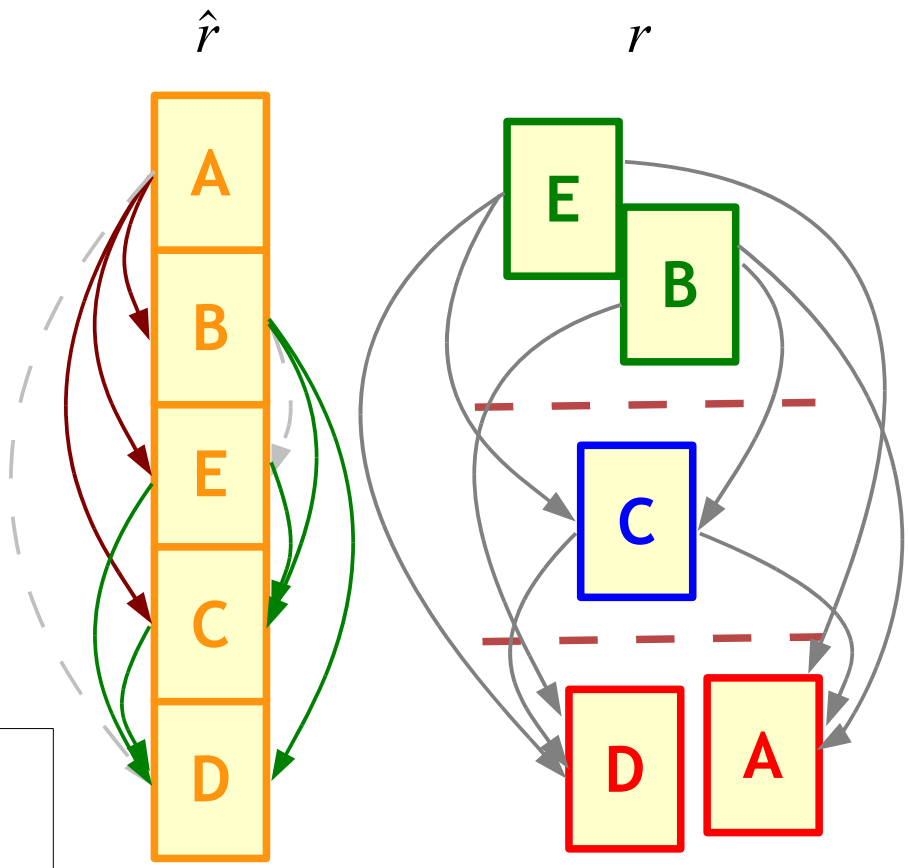
Evaluating Multipartite Rankings

- Multipartite rankings:
  - like Bipartite rankings
  - but the target ranking r consists of *multiple* relevance levels $L = \{1 \ldots l\}$, where $l < c$
  - total ranking is a special case where each level has exactly one item

- # of preferences $= \sum_{(i,j)} c_i \cdot c_j \leq \frac{c^2}{2} \cdot (1 - \frac{1}{l})$
  - c_i is the number of items in level I

- C-Index [Gnen & Heller, 2005]
  - straight-forward generalization of AUC
  - fraction of pairs (x_i, x_j) for which

$$l(i) > l(j) \wedge \hat{r}(x_i) < \hat{r}(x_j)$$



$$D_\tau(r, \hat{r}) = 3$$

$$\text{C-Index}(r, \hat{r}) = \frac{5}{8}$$

Evaluating Multipartite Rankings

C-Index

- the C-index can be rewritten as a weighted sum of pairwise AUCs:

$$\text{C-Index}(r,\hat{r}) = \frac{1}{\sum_{i,j>i} c_i \cdot c_j} \sum_{i,j<i} c_i \cdot c_j \cdot \text{AUC}(r_{i,j}, \hat{r}_{i,j})$$

where $r_{i,j}$ and $\hat{r}_{i,j}$ are the rankings r and $\hat{r}$ restricted to levels i and j.

Jonckheere-Terpstra statistic

- is an *unweighted* sum of pairwise AUCs:

$$\text{m-AUC} = \frac{2}{l \cdot (l-1)} \sum_{i,j>i} \text{AUC}(r_{i,j}, \hat{r}_{i,j})$$

- equivalent to well-known multi-class extension of AUC [Hand & Till, MLJ 2001]

Note:
C-Index and m-AUC can be optimized by optimization of pairwise AUCs

Normalized Discounted Cumulative Gain

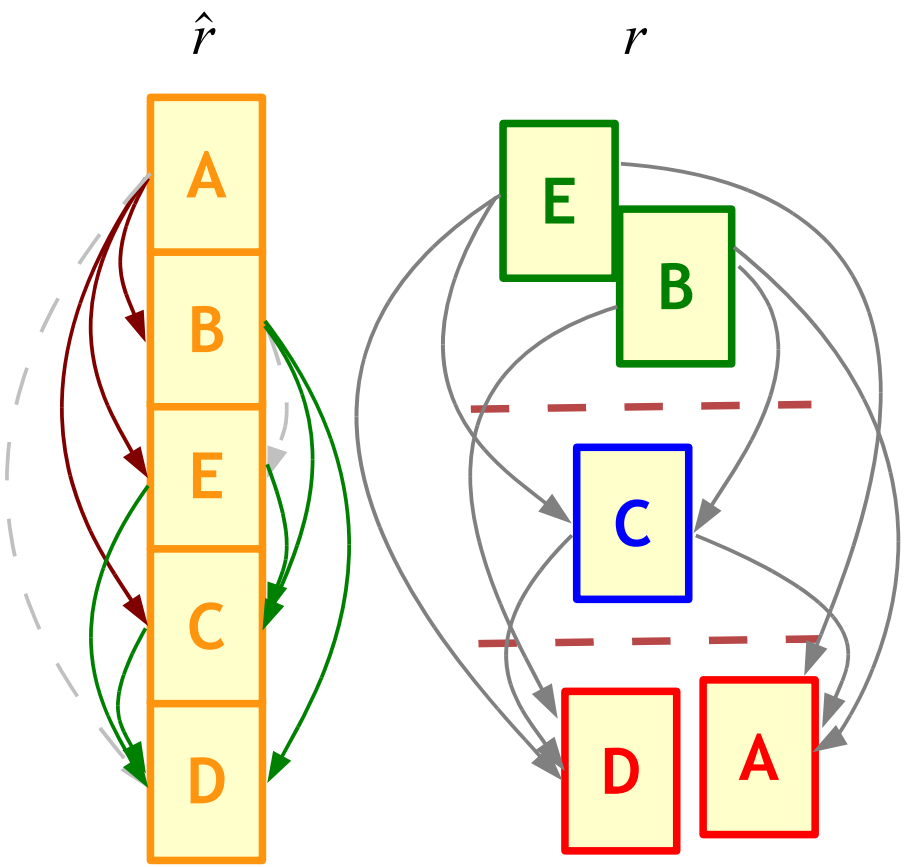
[Jarvelin & Kekalainen, 2002]

- The original formulation of (normalized) discounted cumulative gain refers to this setting

$$DCG(r, \hat{r}) = \sum_{i=1}^{c} \frac{l(i)}{\log(i+1)}$$

  - the sum of the true (relevance) levels of the items
  - each item weighted by its rank in the predicted ranking

- Examples:
  - retrieval of relevant or irrelevant pages
    - 2 relevance levels
  - movie recommendation
    - 5 relevance levels



AGENDA

1. Preference Learning Tasks (Eyke)
2. **Loss Functions** (Johannes)
    a. Evaluation of Rankings
    b. Weighted Measures
    c. Evaluation of Bipartite Rankings
    d. Evaluation of Partial Rankings
3. Preference Learning Techniques (Eyke)
4. Complexity (Johannes)
5. Conclusions

Evaluating Partial Structures in the Predicted Ranking

- For fixed types of partial structures, we have conventional measures
  - bipartite graphs → binary classification
    - accuracy, recall, precision, F1, etc.
    - can also be used when the items are labels!
      - e.g., accuracy on the set of labels for multilabel classification
  - multipartite graphs → ordinal classification
    - multiclass classification measures (accuracy, error, etc.)
    - regression measures (sum of squared errors, etc.)

- For general partial structures
  - some measures can be directly used on the reduced set of target preferences
    - Kendall's distance, Gamma coefficient
  - we can also use set measures on the set of binary preferences
    - both, the source and the target ranking consist of a set of binary preferences
    - e.g. Jaccard Coefficient
      - size of interesection over size of union of the binary preferences in both sets

Gamma Coefficient

- Key idea: normalized difference between
  - number of correctly ranked pairs (Kendall's distance)

    $d = D_\tau(r, \hat{r})$

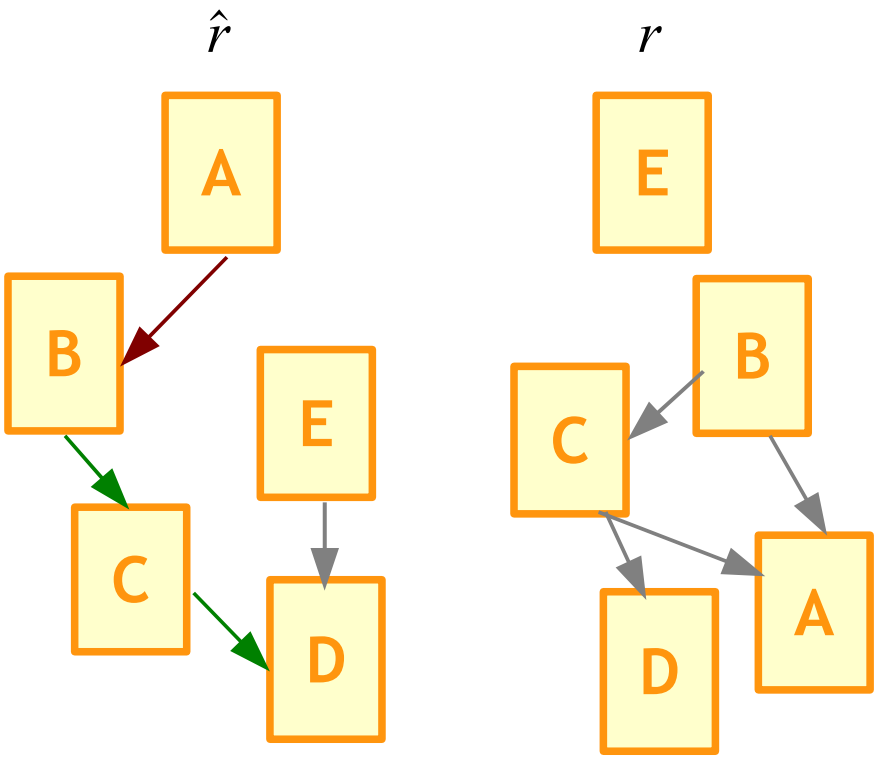
  - number of incorrectly ranked pairs

    $\bar{d} = |\{(i,j) \mid r(x_i) < r(x_j) \wedge \hat{r}(x_i) < \hat{r}(x_j)\}|$

- Gamma Coefficient
  [Goodman & Kruskal, 1979]

  $$\gamma(r, \hat{r}) = \frac{d - \bar{d}}{d + \bar{d}} \in [-1, +1]$$

  - Identical to Kendall's tau if both rankings are total
    - i.e., if $d + \bar{d} = \frac{c \cdot (c-1)}{2}$



$$\gamma(r, \hat{r}) = \frac{2-1}{2+1} = \frac{1}{3}$$

References

- Cheng W., Rademaker M., De Baets B., Hüllermeier E.: *Predicting Partial Orders: Ranking with Abstention*. Proceedings ECML/PKDD-10(1): 215-230 (2010)
- Fürnkranz J., Hüllermeier E., Vanderlooy S.: *Binary Decomposition Methods for Multipartite Ranking***.** Proceedings ECML/PKDD-09(1): 359-374 (2009)
- Gnen M., Heller G.: *Concordance probability and discriminatory power in proportional hazards regression*. Biometrika **92**(4):965–970 (2005)
- Goodman, L., Kruskal, W.: *Measures of Association for Cross Classifications*. Springer-Verlag, New York (1979)
- Hand D.J., Till R.J.: *A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems*. Machine Learning **45**(2):171-186 (2001)
- Jarvelin K., Kekalainen J.: *Cumulated gain-based evaluation of IR techniques.* ACM Transactions on Information Systems **20**(4): 422–446 (2002)
- Jonckheere, A. R.: *A distribution-free k-sample test against ordered alternatives*. Biometrika: 133–145 (1954)
- Kendall, M. *A New Measure of Rank Correlation*. Biometrika 30 (1-2): 81–89 (1938)
- Mann H. B.,Whitney D. R. *On a test of whether one of two random variables is stochastically larger than the other.* Annals of Mathematical Statistics, **18**:50–60 (1947)
- Spearman C. *The proof and measurement of association between two things.* American Journal of Psychology, **15**:72–101 (1904)

AGENDA

1. Preference Learning Tasks (Eyke)
2. Loss Functions (Johannes)
3. **Preference Learning Techniques** (Eyke)
    a. Learning Utility Functions
    b. Learning Preference Relations
    c. Structured Output Prediction
    d. Model-Based Preference Learning
    e. Local Preference Aggregation
4. Complexity of Preference Learning (Johannes)
5. Conclusions

Two Ways of Representing Preferences

- **Utility-based approach:** Evaluating single alternatives

$$U : \mathcal{A} \longrightarrow \mathbb{R}$$

- **Relational approach:** Comparing pairs of alternatives

$$a \succeq b \quad \Leftrightarrow \quad a \text{ is not worse than } b$$

weak preference

$$a \succ b \quad \Leftrightarrow \quad (a \succeq b) \wedge (b \not\succeq a)$$

strict preference

$$a \sim b \quad \Leftrightarrow \quad (a \succeq b) \wedge (b \succeq a)$$

indifference

$$a \perp b \quad \Leftrightarrow \quad (a \not\succeq b) \wedge (b \not\succeq a)$$

incomparability

Utility Functions

- A **utility function** assigns a utility degree (typically a real number or an ordinal degree) to each alternative.
- Learning such a function essentially comes down to solving an (ordinal) **regression problem.**
- Often **additional conditions**, e.g., due to bounded utility ranges or monotonicity properties (→ *learning monotone models*)

- A **utility function induces a ranking** (total order), but not the other way around!
- But it can not represent a **partial order**!
- The **feedback** can be **direct** (exemplary utility degrees given) or **indirect** (inequality induced by order relation):

$$(\boldsymbol{x}, u) \;\Rightarrow\; U(\boldsymbol{x}) \approx u, \qquad \boldsymbol{x} \succ \boldsymbol{y} \;\Leftrightarrow\; U(\boldsymbol{x}) > U(\boldsymbol{y})$$

direct feedback — indirect feedback

Predicting Utilities on Ordinal Scales

(Graded) multilabel classification

X1	X2	X3	X4	A	B	C	D
0.34	0	10	174	--	+	++	0
1.45	0	32	277	0	++	--	+
1.22	1	46	421	--	--	0	+
0.74	1	25	165	0	+	+	++
0.95	1	72	273	+	0	++	--
1.04	0	33	158	+	+	++	--

Collaborative filtering

	P1	P2	P3	...	P38	...	P88	P89	P90
U1	1		4	...		...		3	
U2		2	2	...		...	1		
...				...		...			
U46	?	2	?	...	?	...	?	?	4
...				...		...			
U98	5			...		...	4		
U99			1	...		...		2	

Exploiting dependencies (correlations) between items (labels, products, …).

→ see work in MLC and RecSys communities

Learning Utility Functions from Indirect Feedback

- A (latent) utility function can also be used to solve ranking problems, such as instance, object or label ranking
  → **ranking by (estimated) utility degrees (scores)**

Object ranking

$$(0.74, 1, 25, 165) \succ (0.45, 0, 35, 155)$$
$$(0.47, 1, 46, 183) \succ (0.57, 1, 61, 177)$$
$$(0.25, 0, 26, 199) \succ (0.73, 0, 46, 185)$$
$$(0.95, 0, 73, 133) \succ (0.25, 1, 35, 153)$$
$$(0.68, 1, 55, 147) \succ (0.67, 0, 63, 182)$$

→

Find a utility function that agrees as much as possible with the preference information in the sense that, for most examples,

$$\boldsymbol{x}_i \succ \boldsymbol{y}_i \quad \Leftrightarrow \quad U(\boldsymbol{x}_i) > U(\boldsymbol{y}_i)$$

Instance ranking

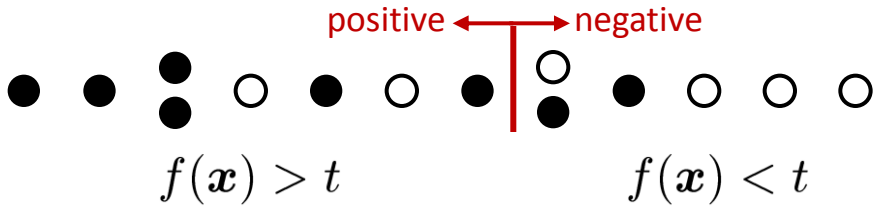
X1	X2	X3	X4	class
0.34	0	10	174	--
1.45	0	32	277	0
1.22	1	46	421	--
0.74	1	25	165	++
0.95	1	72	273	+

→

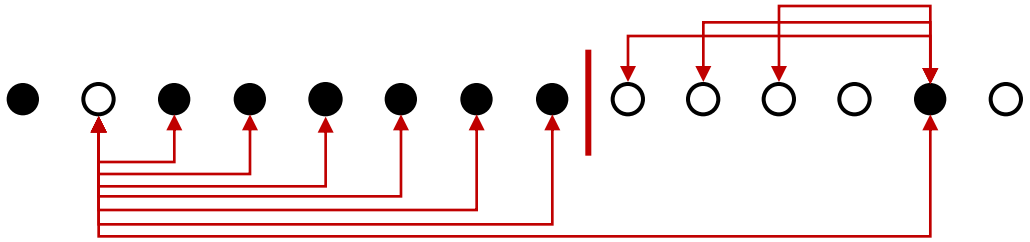
Absolute preferences given, so in principle an ordinal regression problem. However, the goal is to maximize ranking instead of classification performance.

Ranking versus Classification

A ranker can be turned into a classifier via thresholding:



A good classifier is not necessarily a good ranker:



2 classification but
10 ranking errors

→ *learning **AUC-optimizing** scoring classifiers* !

RankSVM and Related Methods (Bipartite Case)

- The idea is to minimize a convex upper bound on the empirical ranking error over a class of (kernalized) ranking functions:

regularizer

$$f^* \in \arg\min_{f \in \mathcal{F}} \left\{ \frac{1}{|P| \cdot |N|} \sum_{\boldsymbol{x} \in P} \sum_{\boldsymbol{x}' \in N} L(f, \boldsymbol{x}, \boldsymbol{x}') + \lambda \cdot R(f) \right\}$$

convex upper bound on

$$\mathbb{I}\left(f(\boldsymbol{x}) < f(\boldsymbol{x}')\right)$$

RankSVM and Related Methods (Bipartite Case)

- The bipartite RankSVM algorithm [Herbrich et al. 2000, Joachimes 2002]:

regularizer

$$f^* \in \arg\min_{f \in \mathcal{F}_K} \left\{ \frac{1}{|P| \cdot |N|} \sum_{\boldsymbol{x} \in P} \sum_{\boldsymbol{x}' \in N} \left(1 - (f(\boldsymbol{x}) - f(\boldsymbol{x}'))_+ + \frac{\lambda}{2} \cdot \|f\|_K^2 \right\}$$

hinge loss

reproducing kernel Hilbert space (RKHS) with kernel $\mathbf{K}$

→ learning comes down to solving a QP problem

RankSVM and Related Methods (Bipartite Case)

- The bipartite RankBoost algorithm [Freund et al. 2003]:

$$f^* \in \arg \min_{f \in \mathcal{L}(\mathcal{F}_{base})} \left\{ \frac{1}{|P| \cdot |N|} \sum_{\boldsymbol{x} \in P} \sum_{\boldsymbol{x}' \in N} \exp\left(-(f(\boldsymbol{x}) - f(\boldsymbol{x}'))\right) \right\}$$

class of linear combinations of base functions

→ learning by means of boosting techniques

Learning Utility Functions for Label Ranking

Label ranking is the problem of learning a function $\mathcal{X} \longrightarrow \Omega$, with Ω the set of rankings (permutations) of a label set $\mathcal{Y} = \{y_1, y_2, \ldots, y_k\}$, from exemplary pairwise preferences $y_i \succ_{\boldsymbol{x}} y_j$.

Can be tackled by learning utility functions $U_1(\cdot), \ldots, U_k(\cdot)$ that are as much as possible (but not too much) in agreement with the preferences in the training data. Given a new query $\boldsymbol{x}$, the labels are ranked according to utility degrees, i.e., a permutation π is predicted such that

$$U_{\pi^{-1}(1)}(\boldsymbol{x}) > U_{\pi^{-1}(2)}(\boldsymbol{x}) > \ldots > U_{\pi^{-1}(k)}(\boldsymbol{x})$$

Label Ranking: Reduction to Binary Classification [Har-Peled et al. 2002]

Proceeding from linear utility functions

$$U_i(\boldsymbol{x}) = \boldsymbol{w}_i \times \boldsymbol{x} = (w_{i,1}, w_{i,2}, \ldots, w_{i,m})(x_1, x_2, \ldots, x_m)^\top,$$

a binary preference $y_i \succ_{\boldsymbol{x}} y_j$ is equivalent to

$$U_i(\boldsymbol{x}) > U_j(\boldsymbol{x}) \Leftrightarrow \boldsymbol{w}_i \times \boldsymbol{x} > \boldsymbol{w}_j \times \boldsymbol{x} \Leftrightarrow (\boldsymbol{w}_i - \boldsymbol{w}_j) \times \boldsymbol{x} > 0$$

and can be modeled as a linear constraint

$$\underbrace{(\boldsymbol{w}_1, \boldsymbol{w}_2 \ldots \boldsymbol{w}_k)}_{\text{(m x k)-dimensional weight vector}} \times \underbrace{(0 \ldots 0\ \boldsymbol{x}\ 0 \ldots 0\ -\boldsymbol{x}\ 0 \ldots 0)^\top}_{\text{positive example in the new instance space}} > 0$$

→ each **pairwise comparison** is turned into a **binary classification** example in a high-dimensional space!

AGENDA

1. Preference Learning Tasks (Eyke)
2. Loss Functions (Johannes)
3. Preference Learning Techniques (Eyke)
    a. Learning Utility Functions
    b. Learning Preference Relations
    c. Structured Output Prediction
    d. Model-Based Preference Learning
    e. Local Preference Aggregation
4. Complexity of Preference Learning (Johannes)
5. Conclusions

Learning Binary Preference Relations

- Learning **binary preferences** (in the form of predicates $\mathrm{P}(\mathbf{x},\mathbf{y})$) is often simpler, especially if the training information is given in this form, too.
- However, it implies an additional step, namely **extracting a ranking** from a (predicted) preference relation.
- This step is not always trivial, since a predicted preference relation may exhibit inconsistencies and may not suggest a unique ranking in an unequivocal way.

instance $\boldsymbol{x}$ $\longrightarrow$

$f_{i,j}$	y_1	y_2	y_3	y_4	y_5
y_1		1	1	0	0
y_2	0		0	1	0
y_3	0	1		0	0
y_4	1	0	1		1
y_5	1	1	1	0	

inference $\longrightarrow$ $y_4 \succ y_5 \succ y_1 \succ y_3 \succ y_2$

Object Ranking: Learning to Order Things [Cohen et al. 99]

- In a first step, a **binary preference function** $\mathbf{PREF}$ is constructed; $\mathbf{PREF}(\mathbf{x},\mathbf{y}) \in [0,1]$ is a measure of the certainty that $\mathbf{x}$ should be ranked before $\mathbf{y}$, and $\mathbf{PREF}(\mathbf{x},\mathbf{y})=1-\mathbf{PREF}(\mathbf{y},\mathbf{x})$.
- This function is expressed as a linear combination of base preference functions:

$$\mathrm{PREF}(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{N} w_i \cdot R_i(\boldsymbol{x},\boldsymbol{y})$$

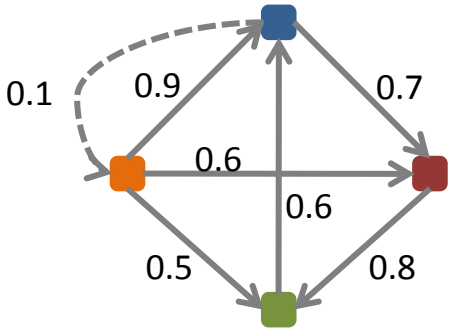
- The weights can be learned, e.g., by means of the weighted majority algorithm [Littlestone & Warmuth 94].
- In a second step, a total order is derived, which is a much as possible in agreement with the binary preference relation.

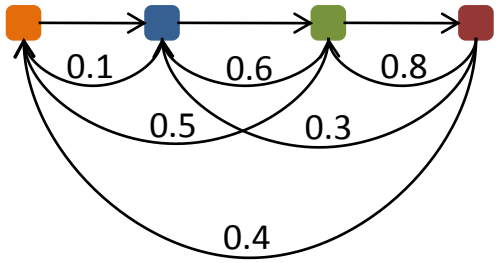
Object Ranking: Learning to Order Things [Cohen et al. 99]

- The weighted feedback arc set problem: Find a permutation π such that

$$\sum_{(x,y):\pi(x)>\pi(y)} \mathrm{PREF}(x,y)$$

becomes minimal.





cost = 0.1+0.6+0.8+0.5+0.3+0.4 = 2.7

Object Ranking: Learning to Order Things [Cohen et al. 99]

- Since this is an NP-hard problem, it is solved heuristically.

Input: an instance set X; a preference function PREF
Output: an approximately optimal ordering function $\hat{\rho}$
let $V = X$
for each $v \in V$ ***do***
while V is non-empty ***do*** $\pi(v) = \sum_{u \in V} \mathrm{PREF}(v,u) - \sum_{u \in V} \mathrm{PREF}(u,v)$
    let $t = \arg\max_{u \in V} \pi(u)$
    let $\hat{\rho}(t) = |V|$
        $V = V - \{t\}$
    for each $v \in V$ ***do*** $\pi(v) = \pi(v) + \mathrm{PREF}(t,v) - \mathrm{PREF}(v,t)$
endwhile

- The algorithm successively chooses nodes having **maximal „net-flow“** within the remaining subgraph.
- It can be shown to provide a **2-approximation** to the optimal solution.

Label Ranking: Learning by Pairwise Comparison (LPC) [Hüllermeier et al. 2008]

Label ranking is the problem of learning a function $\mathcal{X} \longrightarrow \Omega$, with Ω the set of rankings (permutations) of a label set $\mathcal{Y} = \{y_1, y_2, \ldots, y_k\}$, from exemplary pairwise preferences $y_i \succ_{\boldsymbol{x}} y_j$.

LPC trains a model

$$\mathcal{M}_{i,j} : \mathcal{X} \longrightarrow [0, 1]$$

for all $i < j$. Given a query instance $\boldsymbol{x}$, this model is supposed to predict whether $y_i \succ y_j$ $(\mathcal{M}_{i,j}(\boldsymbol{x}) = 1)$ or $y_j \succ y_i$ $(\mathcal{M}_{i,j}(\boldsymbol{x}) = 0)$.

More generally, $\mathcal{M}_{i,j}(\boldsymbol{x})$ is the estimated probability that $y_i \succ y_j$.

Decomposition into $k(k-1)/2$ **binary classification problems**.

Label Ranking: Learning by Pairwise Comparison (LPC) [Hüllermeier et al. 2008]

Training data (for the label pair A and B):

X1	X2	X3	X4	preferences	class
0.34	0	10	174	**A** $\succ$ **B**, B $\succ$ C, C $\succ$ D	1
~~1.45~~	~~0~~	~~32~~	~~277~~	~~B $\succ$ C~~	
1.22	1	46	421	B $\succ$ D, **B** $\succ$ **A**, C $\succ$ D, A $\succ$ C	0
0.74	1	25	165	C $\succ$ A, C $\succ$ D, **A** $\succ$ **B**	1
~~0.95~~	~~1~~	~~72~~	~~273~~	~~B $\succ$ D, A $\succ$ D,~~	
1.04	0	33	158	D $\succ$ A, **A** $\succ$ **B**, C $\succ$ B, A $\succ$ C	1

Label Ranking: Learning by Pairwise Comparison (LPC) [Hüllermeier et al. 2008]

At prediction time, a query instance is submitted to all models, and the predictions are combined into a binary preference relation:

predictions $\mathcal{M}_{i,j}(\boldsymbol{x})$ →

	A	B	C	D
A		0.3	0.8	0.4
B	0.7		0.7	0.9
C	0.2	0.3		0.3
D	0.6	0.1	0.7	

Label Ranking: Learning by Pairwise Comparison (LPC) [Hüllermeier et al. 2008]

At prediction time, a query instance is submitted to all models, and the predictions are combined into a binary preference relation:

predictions $\mathcal{M}_{i,j}(\boldsymbol{x})$ →

	A	B	C	D	
A		0.3	0.8	0.4	1.5
B	0.7		0.7	0.9	2.3
C	0.2	0.3		0.3	0.8
D	0.6	0.1	0.7		1.4

$B \succ A \succ D \succ C$

From this relation, a ranking is derived by means of a **ranking procedure**. In the simplest case, this is done by sorting the labels according to their sum of **weighted votes**.

AGENDA

1. Preference Learning Tasks (Eyke)
2. Loss Functions (Johannes)
3. Preference Learning Techniques (Eyke)
    a. Learning Utility Functions
    b. Learning Preference Relations
    c. Structured Output Prediction
    d. Model-Based Preference Learning
    e. Local Preference Aggregation
4. Complexity of Preference Learning (Johannes)
5. Conclusions

Structured Output Prediction [Bakir et al. 2007]

- Rankings, multilabel classifications, etc. can be seen as specific types of **structured** (as opposed to scalar) **outputs**.
- Discriminative structured prediction algorithms infer a **joint scoring function on input-output pairs** and, for a given input, predict the output that maximises this scoring function.
- Joint feature map and scoring function

$$\phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d, \quad f(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) = \langle \boldsymbol{w}, \phi(\boldsymbol{x}, \boldsymbol{y}) \rangle$$

- The learning problem consists of estimating the weight vector, e.g., using structural risk minimization.
- Prediction requires solving a **decoding problem**:

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y} \in \mathcal{Y}} f(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) = \arg\max_{\boldsymbol{y} \in \mathcal{Y}} \langle \boldsymbol{w}, \phi(\boldsymbol{x}, \boldsymbol{y}) \rangle$$

Structured Output Prediction [Bakir et al. 2007]

- **Preferences** are expressed through **inequalities** on inner products:

$$\min_{\boldsymbol{w},\xi} \; \|\|w\|\|^2 + \nu \sum_{i=1}^{m} \xi_i$$

loss function

$$\text{s.t.} \quad \langle \boldsymbol{w}, \phi(\boldsymbol{x}_i, \boldsymbol{y}_i) \rangle - \langle \boldsymbol{w}, \phi(\boldsymbol{x}_i, \boldsymbol{y}) \rangle \geq \Delta(\boldsymbol{y}_i, \boldsymbol{y}) - \xi_i \text{ for all } \boldsymbol{y} \in \mathcal{Y}$$

$$\xi_i \geq 0 \quad (i = 1, \ldots, m)$$

- The potentially huge **number of constraints** cannot be handled explicitly and calls for specific techniques (such as cutting plane optimization)

AGENDA

1. Preference Learning Tasks (Eyke)
2. Loss Functions (Johannes)
3. Preference Learning Techniques (Eyke)
    a. Learning Utility Functions
    b. Learning Preference Relations
    c. Structured Output Prediction
    d. Model-Based Preference Learning
    e. Local Preference Aggregation
4. Complexity of Preference Learning (Johannes)
5. Conclusions

Model-Based Methods for Ranking

- **Model-based approaches** to ranking proceed from specific assumptions about the possible rankings (**representation bias**) or make use of **probabilistic models** for rankings (parametrized probability distributions on the set of rankings).

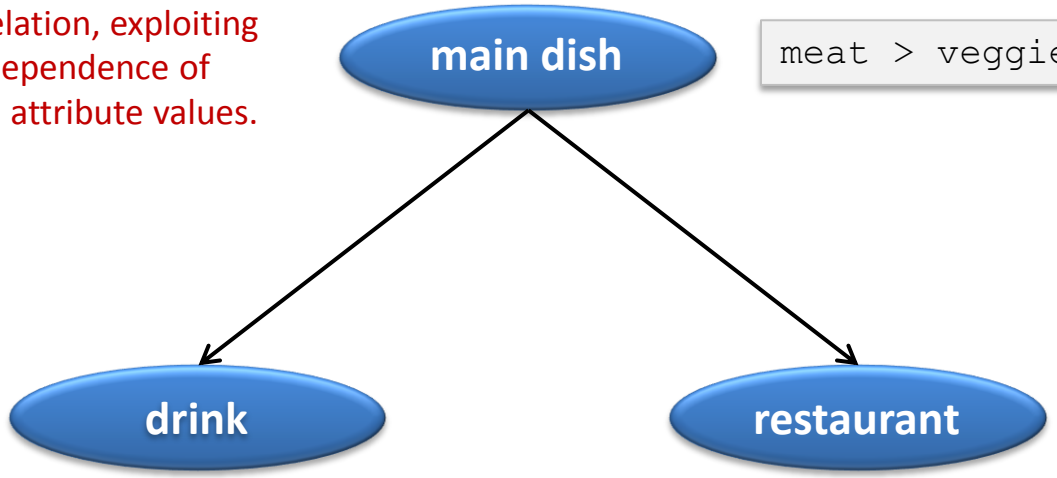
- In the following, we shall see examples of both type:
  - Restriction to lexicographic preferences
  - Conditional preference networks (CP-nets)
  - Label ranking using the Plackett-Luce model

Learning Lexicographic Preference Models [Yaman et al. 2008]

- Suppose that objects are represented as feature vectors of length m, and that each attribute has k values.
- For $\mathrm{n}=\mathrm{k}^{\mathrm{m}}$ objects, there are $\mathrm{n}!$ permutations (rankings).
- A **lexicographic order** is uniquely determined by
  - a total order of the attributes
  - a total order of each attribute domain
- **Example:** Four binary attributes ($\mathrm{m}=4$, $\mathrm{k}=2$)
  - there are $16! \approx 2 \cdot 10^{13}$ rankings
  - but only $(2^4) \cdot 4! = 384$ of them can be expressed in terms of a lexicographic order
- [Yaman et al. 2008] present a learning algorithm that explictly maintains the version space, i.e., the attribute-orders compatible with all pairwise preferences seen so far (assuming binary attributes with 1 preferred to 0). Predictions are derived based on the „votes“ of the consistent models.

Learning Conditional Preference (CP) Networks [Chevaleyre et al. 2010]

Compact representation of a partial order relation, exploiting conditional independence of preferences on attribute values.




```
meat > veggie > fish
```

```
meat:    red wine    > white wine
veggie:  red wine    > white wine
fish:    white wine  > red wine
```

```
meat:    Italian > Chinese
veggie:  Chinese > Italian
fish:    Chinese > Italian
```

Training data (possibly noisy):

```
(meat, red wine, Italian)         > (veggie, red wine, Italian)
(fish, whited wine, Chinease)     > (veggie, red wine, Chinease)
(veggie, whited wine, Chinease)   > (veggie, red wine, Italian)
            …                     …                  …
```

Label Ranking based on the Plackett-Luce Model [Cheng et al. 2010c]

The Plackett-Luce (PL) model is specified by a parameter vector $\boldsymbol{v} = (v_1, v_2, \ldots v_m) \in \mathbb{R}_+^m$:

$$\mathbf{P}(\pi \,|\, \boldsymbol{v}) = \prod_{i=1}^{m} \frac{v_{\pi(i)}}{v_{\pi(i)} + v_{\pi(i+1)} + \ldots + v_{\pi(m)}}$$

Reduces problem to learning a mapping $\boldsymbol{x} \mapsto \boldsymbol{v}$.

Example: $\boldsymbol{v} = (1, 4, 2)$, $\quad \mathbf{P}(\pi \,|\, \boldsymbol{v}) = \frac{v_{\pi(1)}}{v_{\pi(1)}+v_{\pi(2)}+v_{\pi(3)}} \cdot \frac{v_{\pi(2)}}{v_{\pi(2)}+v_{\pi(3)}} \cdot 1$

1	2	3	0.0952
1	3	2	0.0476
2	1	3	0.1905
2	3	1	0.0571
3	1	2	0.3810
3	2	1	0.2286

ML Estimation of the Weight Vector in Label Ranking

Assume $\boldsymbol{x} = (x_1, \ldots, x_D) \in \mathbb{R}^D$ and model the v_i as log-linear functions:

$$v_i \;=\; \exp\left(\sum_{d=1}^{D} \alpha_d^{(i)} \cdot x_d\right)$$

← can be seen as a log-linear utility function of i-th label

Given training data $\mathcal{T} = \left\{\left(\boldsymbol{x}^{(n)}, \pi^{(n)}\right)\right\}_{n=1}^{N}$ with $\boldsymbol{x}^{(n)} = \left(x_1^{(n)}, \ldots, x_D^{(n)}\right)$, the log-likelihood is given by

$$L = \sum_{n=1}^{N} \left[\sum_{m=1}^{M_n} \log\left(v(\pi^{(n)}(m), n)\right) - \log \sum_{j=m}^{M_n} v(\pi^{(n)}(j), n)\right],$$

convex function, maximization through gradient ascent

where M_n is the number of labels in the ranking $\pi^{(n)}$, and

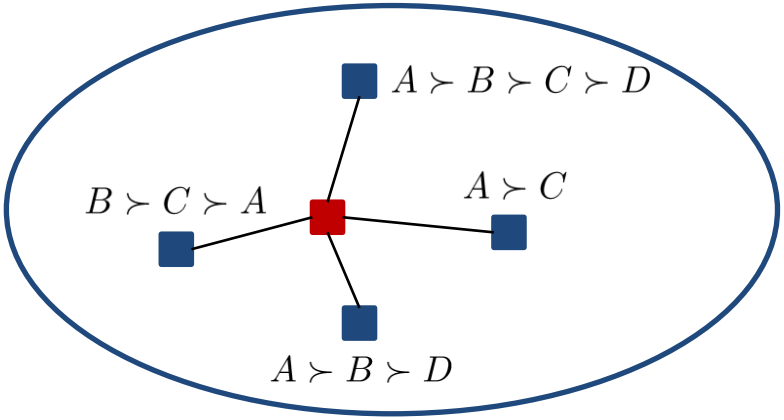
$$v(m, n) \;=\; \exp\left(\sum_{d=1}^{D} \alpha_d^{(m)} \cdot x_d^{(n)}\right) \quad .$$

AGENDA

1. Preference Learning Tasks (Eyke)
2. Loss Functions (Johannes)
3. Preference Learning Techniques (Eyke)
    a. Learning Utility Functions
    b. Learning Preference Relations
    c. Structured Output Prediction
    d. Model-Based Preference Learning
    e. Local Preference Aggregation
4. Complexity of Preference Learning (Johannes)
5. Conclusions

Learning Local Preference Models [Cheng et al. 2009]

- Main idea of **instance-based (lazy) learning**: Given a new query (instance for which a prediction is requested), search for similar instances in a „case base“ (stored examples) and combine their outputs into a prediction.
- This is especially appealing for predicting **structured outputs** (like rankings) in a complex space $\mathcal{Y}$, as it circumvents the construction and explicit representation of a „$\mathcal{Y}$-valued“ function.
- In the case of **ranking**, it essentially comes down to **aggregating** a set of (possibly partial or incomplete) rankings.



Learning Local Preference Models: Rank Aggregation

- Finding the generalized median:

$$\hat{\boldsymbol{y}} = \arg\min_{\boldsymbol{y}\in\mathcal{Y}} \sum_{i=1}^{k} \Delta(\boldsymbol{y}_i, \boldsymbol{y})$$

- If **Kendall's tau** is used as a distance, the generalized median is called the **Kemendy-optimal ranking**. Finding this ranking is an NP-hard problem (weighted feedback arc set tournament).
- In the case of **Spearman's rho** (sum of squared rank distances), the problem can easily be solved through Borda count.

Learning Local Preference Models: Probabilistic Estimation

- Another approach is to assume the neighbored rankings to be generated by a **locally constant probability distribution**, to estimate the parameters of this distribution, and then to predict the mode [Cheng et al. 2009].
- For example, using again the PL model:

$$\mathbf{P}(\pi_1, \ldots, \pi_k \,|\, \boldsymbol{v}) = \prod_{j=1}^{k} \mathbf{P}(\pi_j \,|\, \boldsymbol{v}) = \prod_{j=1}^{k} \prod_{i=1}^{m} \frac{v_{\pi(i)}}{v_{\pi(i)} + v_{\pi(i+1)} + \ldots + v_{\pi(m)}}$$

$$\log L = \sum_{j=1}^{k} \sum_{i=1}^{m} \log\left(v_{\pi(i)}\right) - \log\left(v_{\pi(i)} + v_{\pi(i+1)} + \ldots + v_{\pi(m)}\right)$$

- Can easily be generalized to the case of incomplete rankings [Cheng et al. 2010c].

Summary of Main Algorithmic Principles

- **Reduction** of ranking to (binary) classification (e.g., constraint classification, LPC)
- **Direct optimization** of (regularized) smooth approximation of ranking losses (RankSVM, RankBoost, …)
- **Structured output prediction**, learning joint scoring („matching“) function
- Learning parametrized **statistical ranking models** (e.g., Plackett-Luce)
- **Restricted model classes**, fitting (parametrized) deterministic models (e.g., lexicographic orders)
- **Lazy learning**, local preference aggregation (lazy learning)

References

- G. Bakir, T. Hofmann, B. Schölkopf, A. Smola, B. Taskar and S. Vishwanathan. *Predicting structured data*. MIT Press, 2007.
- W. Cheng, K. Dembczynski and E. Hüllermeier. *Graded Multilabel Classification: The Ordinal Case*. ICML-2010, Haifa, Israel, 2010.
- W. Cheng, K. Dembczynski and E. Hüllermeier. *Label ranking using the Plackett-Luce model*. ICML-2010, Haifa, Israel, 2010.
- W. Cheng and E. Hüllermeier. *Predicting partial orders: Ranking with abstention*. ECML/PKDD-2010, Barcelona, 2010.
- W. Cheng, C. Hühn and E. Hüllermeier. *Decision tree and instance-based learning for label ranking*. ICML-2009.
- Y. Chevaleyre, F. Koriche, J. Lang, J. Mengin, B. Zanuttini. *Learning ordinal preferences on multiattribute domains: The case of CP-nets*. In: J. Fürnkranz and E. Hüllermeier (eds.) Preference Learning, Springer-Verlag, 2010.
- W.W. Cohen, R.E. Schapire and Y. Singer. *Learning to order things*. Journal of Artificial Intelligence Research, 10:243–270, 1999.
- Y. Freund, R. Iyer, R. E. Schapire and Y. Singer. *An efficient boosting algorithm for combining preferences*. Journal of Machine Learning Research, 4:933–969, 2003.
- J. Fürnkranz, E. Hüllermeier, E. Mencia, and K. Brinker. *Multilabel Classification via Calibrated Label Ranking*. Machine Learning 73(2):133-153, 2008.
- J. Fürnkranz, E. Hüllermeier and S. Vanderlooy. *Binary decomposition methods for multipartite ranking*. Proc. ECML-2009, Bled, Slovenia, 2009.
- D. Goldberg, D. Nichols, B.M. Oki and D. Terry. *Using collaborative filtering to weave and information tapestry*. Communications of the ACM, 35(12):61–70, 1992.
- S. Har-Peled, D. Roth and D. Zimak. *Constraint classification: A new approach to multiclass classification*. Proc. ALT-2002.
- R. Herbrich, T. Graepel and K. Obermayer. *Large margin rank boundaries for ordinal regression*. Advances in Large Margin Classifiers, 2000.
- E. Hüllermeier, J. Fürnkranz, W. Cheng and K. Brinker. *Label ranking by learning pairwise preferences*. Artificial Intelligence, 172:1897–1916, 2008.
- T. Joachims. *Optimizing search engines using clickthrough data*. Proc. KDD 2002.
- N. Littlestone and M.K. Warmuth. *The weighted majority algorithm*. Information and Computation, 108(2): 212–261, 1994.
- G. Tsoumakas and I. Katakis. *Multilabel classification: An overview*. Int. J. Data Warehouse and Mining, 3:1–13, 2007.
- F. Yaman, T. Walsh, M. Littman and M. desJardins. *Democratic Approximation of Lexicographic Preference Models*. ICML-2008.

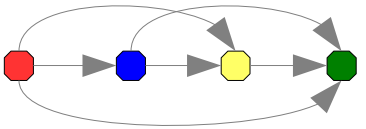
AGENDA

1. Preference Learning Tasks (Eyke)
2. Loss Functions (Johannes)
3. Preference Learning Techniques (Eyke)
4. **Complexity of Preference Learning** (Johannes)
   a. **Training Complexity**
      - SVMRank
      - Pairwise Methods
   b. Prediction Complexity
      - Aggregation of Preference Relations is hard
      - Aggregation Strategies
      - Efficient Aggregation
5. Conclusions

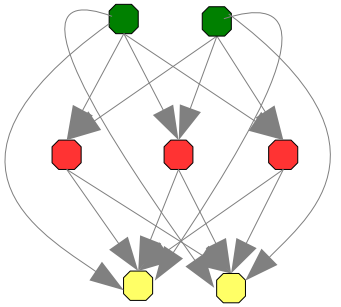
Training Complexity: Number of Preferences

we have d binary preferences for items $X = \{x_1, \ldots, x_c\}$

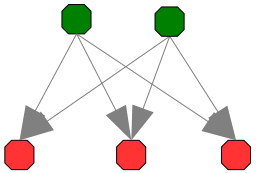
- total ranking: $d = \frac{c \cdot (c-1)}{2}$



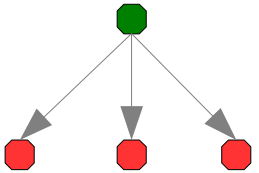
- multi-partite ranking *(k partitions with p_i items each):* $d = \sum_{i \neq j} p_i \cdot p_j$



- bi-partite ranking (with p and c-p items): $d = p \cdot (c-p)$
  (e.g., multi-label classification)



- top rank: $d = c - 1$
  (e.g. classification)



Training Complexity of Relational Approach

We generate one training example for each binary preference

- complexity of the binary base learner is $f(d)$
  - e.g. $f(d)=O(d^2)$ for a learner with quadratic complexity

Single-set ranking:

- We have c items with ranking information
- Total complexity $f(d)$ depends on the density of the ranking information
  - quadratic in c for (almost) full rankings
  - linear in c for bipartite rankings with a constant p

Multi-set ranking:

- We have n sets of c items with ranking information
  - label ranking is a special case of this scenario
  - object ranking where multiple sets of objects are ranked is also a special case
- Total complexity is
  - $f(n \cdot d)$ for approaches where all preferences are learned jointly
    - can be more efficient if f is super-linear and problem is decomposed into smaller subproblems (pairwise label ranking)

Example: Complexity of SVMRank

- Reformulation as Binary SVM [Herbrich et al. 2000, Joachims 2002]
  - d constraints of the form $\boldsymbol{w}^T(\boldsymbol{x}_i - \boldsymbol{x}_j) \geq 1 - \xi_{ij}$
  - d slack variables ξ_{ij}

  Total complexity: $f(d)$

  where $f(.)$ is the complexity for solving the quadratic program
  - super-linear for conventional training algorithms like SMO, SVM-light, etc.

- Reformulation as Structural SVM [Joachims 2006]
  - 2^d constraints of the form $\frac{1}{d} \cdot \boldsymbol{w}^T \sum_{\boldsymbol{x}_i > \boldsymbol{x}_j} c_{ij}(\boldsymbol{x}_i - \boldsymbol{x}_j) \geq \frac{1}{d} \cdot \sum_{\boldsymbol{x}_i > \boldsymbol{x}_j} c_{ij} - \xi$
  - 1 slack variable ξ

  Total complexity: d
  - Cutting-Plane algorithm:
    - iterative algorithm for solving the above problem in linear time
      - iteratively find an appropriate subset of the constraints
      - covergence independent of d
    - further optimization could even yield a total complexity of $\min(n \cdot \log(n), d)$

Example: Complexity of Pairwise Label Ranking

n examples, c classes, d preferences in total, $\bar{d} = \frac{d}{n}$ preferences on average

- decomposed into $\frac{c \cdot (c-1)}{2}$ binary problems
- each problem has n_{ij} examples $\sum_{ij} n_{ij} = d$

→ total training complexity $\sum_{ij} f(n_{ij}) \leq \bar{d} \cdot f(n) \leq f(d) = f\left(\sum_{ij} n_{ij}\right)$

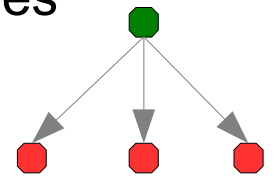
[Hüllermeier et al. 2008]

- upper bounds are tight if f is linear
- big savings are possible super-linear complexities $f(n) = n^o \ (o > 1)$
  - distributing the same number of examples over a larger number of smaller dataset is more efficient

$$o > 1 \rightarrow \sum n_i^o < \left(\sum n_i\right)^o$$

Example: Complexity of Pairwise Classification

- Pairwise classification can be considered as a label ranking problem
  - for each example the correct class is preferred over all other classes



→ Total training complexity $\leq (c-1)\cdot f(n)$

For comparison:

- Constraint Classification:
  - Utility-based approach that learns one theory from all $(c-1)\cdot n$ examples

  Total training complexity: $f\left((c-1)\cdot n\right)$

- One-Vs-All Classification:
  - different class binarization that learns one theory for each class

  Total training complexity: $c\cdot f(n)$

AGENDA

1. Preference Learning Tasks (Eyke)
2. Loss Functions (Johannes)
3. Preference Learning Techniques (Eyke)
4. **Complexity of Preference Learning** (Johannes)
   a. Training Complexity
      - SVMRank
      - Pairwise Methods
   b. Prediction Complexity
      - Aggregation of Preference Relations is hard
      - Aggregation Strategies
      - Efficient Aggregation
5. Conclusions

Prediction Complexity

f complexity for evaluating a single classifier, c items to rank

- Utility-Based Approaches:
  - compute the utilities for each item: $c \cdot f$
  - sort the items according to utility: $c \cdot \log(c)$

$$O\left(c \cdot (\log(c) + f)\right)$$

- Relational Approaches:
  - compute all pairwise predictions: $\frac{c \cdot (c-1)}{2} \cdot f$
  - aggregate them into an overall ranking
    - method-dependent complexity

$$O\left(c^2 \cdot f\right)$$

  - Can we do better?

Aggregation is NP-Hard

- The key problem with aggregation is that the learned preference function may not be transitive.
  - Thus, a total ordering will violate some constraints

Aggregation Problem:

- Find the total order that violates the least number of predicted preferences.

- equivalent to the Feedback Arc Set problem for tournaments
  - What is the minimum number of edges in a directed graph that need to be inverted so that the graph is acyclic?
- This is NP-hard [Alon 2006]
  - but there are approximation algorithms with guarantees
    [Cohen et al. 1999, Balcan et al. 2007, Ailon & Mohri 2008, Mathieu & Schudy, to appear]
  - For example, [Ailon et al. 2008]
    - propose Kwiksort, a straight-forward adaption of Quicksort to the aggregation problem
    - prove that it is a randomized expected 3-approximation algorithm

Aggregating Pairwise Predictions

- Aggregate the predictions $P(\lambda_i > \lambda_j)$ of the binary classifiers into a final ranking by computing a score s_i for each class I

  - Voting: count the number of predictions for each class
    (number of points in a tournament)

$$s_i = \sum_{j=1}^{c} \delta\left\{ P(\lambda_i > \lambda_j) > 0.5 \right\} \qquad \delta\{x\} = \begin{cases} 1 & \text{if } x = \text{true} \\ 0 & \text{if } x = \text{false} \end{cases}$$

  - Weighted Voting: weight the predictions by their probability

$$s_i = \sum_{j=1}^{c} P(\lambda_i > \lambda_j)$$

- General Pairwise Coupling problem [Hastie & Tibshirani 1998; Wu, Lin, Weng 2004]
  - Given $P(\lambda_i > \lambda_j) = P(\lambda_i | \lambda_i, \lambda_j)$ for all i, j
  - Find $P(\lambda_i)$ for all i
  - Can be turned into a system of linear equations

Pairwise Classification & Ranking Loss

[Hüllermeier & Fürnkranz, 2010]

- Weighted Voting optimizes Spearman Rank Correlation
  - assuming that pairwise probabilities are estimated correctly
- Kendall's Tau can in principle be optimized
  - NP-hard (feedback arc set problem)

- Different ways of combining the predictions of the binary classifiers optimize different loss functions
  - without the need for re-training of the binary classifiers!

- However, not all loss functions can be optimized
  - e.g., 0/1 loss for rankings cannot be optimized
  - or in general the probability distribution over the rankings cannot be recovered from pairwise information

Speeding Up Classification Time

- Training is efficient, but pairwise classification still has to
  - store a quadratic number of classifiers in memory
  - query all of them for predicting a class

Key Insight:

- Not all comparisons are needed for determining the winning class
- More precisely:
  - If class X has a total score of s
  - and no other class can achieve an equal score

  → we can predict X even if not all comparisons have been made

Algorithmic idea:

- Keep track of the loss points
- if class with smallest loss has played all games, it is the winner

  → focus on the class with the smallest loss
- Can be easily generalized from voting (win/loss) to weighted voting (e.g., estimated pairwise win probabilities)

Quick Weighted Voting

[Park & Fürnkranz, ECML 2007]

while c_{top} *not determined* **do**
  $c_a \leftarrow$ class $c_i \in K$ with minimal l_i;
  $c_b \leftarrow$ class $c_j \in K \backslash \{c_a\}$ with minimal l_j
  & classifier $C_{a,b}$ has not yet been evaluated;
  if *no* c_b *exists* **then**
    $c_{top} \leftarrow c_a$;
  else
    $v_{ab} \leftarrow \texttt{Evaluate}(C_{a,b})$;
    $l_a \leftarrow l_a + (1 - v_{ab})$;
    $l_b \leftarrow l_b + v_{ab}$;

select class with fewest losses

pair it with unplayed class with fewest losses

we're done if no such class can be found

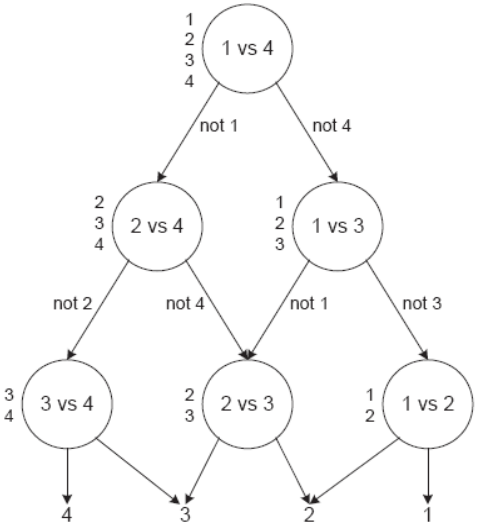
evaluate the classifier and update loss statistics

Decision-Directed Acyclic Graphs

[Platt, Cristianini & Shawe-Taylor, NIPS 2000]

DDAGS

- construct a fixed decoding scheme with $c-1$ decisions
- unclear what loss function is optimized



Comparison to QWeighted

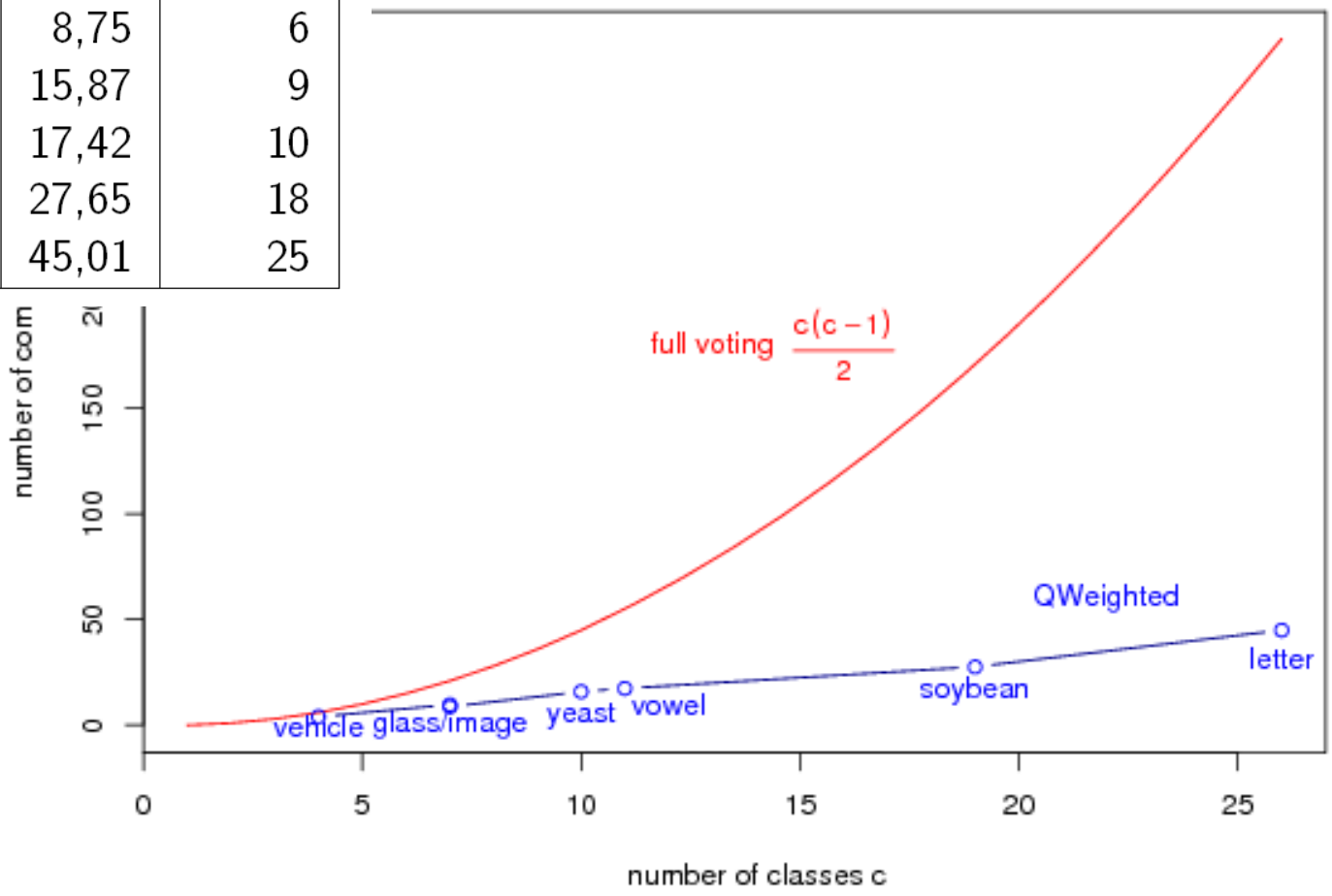
- DDAGs slightly faster
- but considerably less accurate

dataset	JRip		NB		C4.5(J48)		SVM	
vehicle	**73,88**	72,46	**45,39**	44,92	**71,99**	70,92	75,06	75,06
glass	**74,77**	74,30	49,07	49,07	**71,50**	69,16	57,01	**57,94**
image	**96,62**	96,41	80,09	80,09	**96,93**	96,75	93,51	93,51
yeast	**58,96**	58,09	**57,55**	57,21	**58,56**	57,75	**57,68**	57,41
vowel	**82,42**	76,67	**63,84**	63,64	**82,93**	78,28	**81,92**	81,52
soybean	**94,00**	93,56	92,97	92,97	**93,56**	91,80	**94,14**	93,41
letter	**92,33**	88,33	**63,08**	63,00	**91,50**	86,15	**83,80**	82,58

Accuracy : left - QWeighted, right - DDAG

Average Number of Comparisons for QWeighted algorithm

dataset	c	$\frac{c(c-1)}{2}$	QW	DDAG
vehicle	4	6	3,98	3
glass	7	21	9,75	6
image	7	21	8,75	6
yeast	10	45	15,87	9
vowel	11	55	17,42	10
soybean	19	171	27,65	18
letter	26	325	45,01	25



References

- Ailon, N., Charikar, M., and Newman, A. *Aggregating inconsistent information: ranking and clustering*. Journal of the ACM 55, 5, Article 23, 2008.
- Ailon, N. and Mohri, M. *An efficient reduction of ranking to classification*. Procs. 21st COLT-08. 87–97, 2008.
- Alon, N. 2006. *Ranking tournaments.* SIAM J. Discrete Math. 20, 1, 137–142.
- Balcan, M.-F., Bansal, N., Beygelzimer, A., Coppersmith, D., Langford, J., & Sorkin, G. B. *Robust reductions from ranking to classification.* Proceedings COLT-07, pp. 604–619, 2007.
- W. W. Cohen, R. E. Schapire and Y. Singer, *Learning to Order Things*, Journal of AI Research, 10:243-270, 1999.
- J. Fürnkranz: *Round Robin Classification*. Journal of Machine Learning Research 2: 721-747 (2002)
- S. Har-Peled, D. Roth, D. Zimak: *Constraint Classification for Multiclass Classification and Ranking.* Proceedings NIPS 2002: 785-792
- T. Hastie and R. Tibshirani, *Classification by pairwise coupling*, Annals of Statistics 26 (2):451-471, 1998.
- R. Herbrich, T. Graepel, and K. Obermayer. *Large margin rank boundaries for ordinal regression*. In Advances in Large Margin Classifiers, pages 115–132. MIT Press, Cambridge, MA, 2000.
- E. Hüllermeier, J.Fürnkranz, Weiwei Cheng, K. Brinker: *Label ranking by learning pairwise preferences.* Artificial Intelligence 172(16-17): 1897-1916 (2008)
- T. Joachims. *Optimizing search engines using clickthrough data*. In Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD), 2002.
- T. Joachims, *Training Linear SVMs in Linear Time*, Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD), 2006
- C. Mathieu and W. Schudy. *How to Rank with Fewer Errors - A PTAS for Feedback Arc Set in Tournaments,* To appear.
- S.-H. Park, J. Fürnkranz: *Efficient Pairwise Classification.* Proceedings ECML 2007: 658-665
- J. C. Platt, N. Cristianini, J. Shawe-Taylor: *Large Margin DAGs for Multiclass Classification.* Proceedings NIPS 1999: 547-553
- T.-F. Wu, C.-J. Lin and R. C. Weng, *Probability Estimates for Multi-class Classification by Pairwise Coupling,* Journal of Machine Learning Research, 5(975—1005), 2004