AGENDA

1. Preference Learning Tasks
2. Performance Assessment and Loss Functions
3. **Preference Learning Techniques**
    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
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 more general relations, e.g., a **partial order**!
- The **feedback** can be **direct** (exemplary utility degrees given) or **indirect** (inequality induced by order relation):

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

absolute feedback relative 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

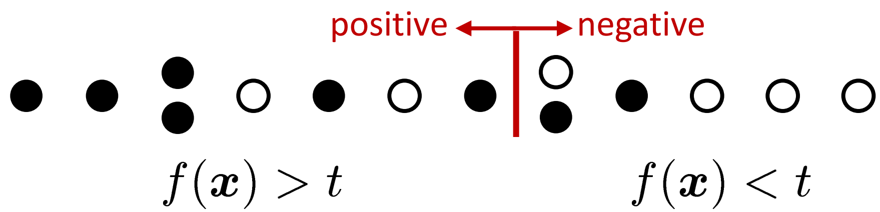
X1	X2	X3	X4	class
0.34	0	10	174	--
1.45	0	32	277	0
1.22	1	46	421	--
0.74	1	25	165	++
0.95	1	72	273	+

→

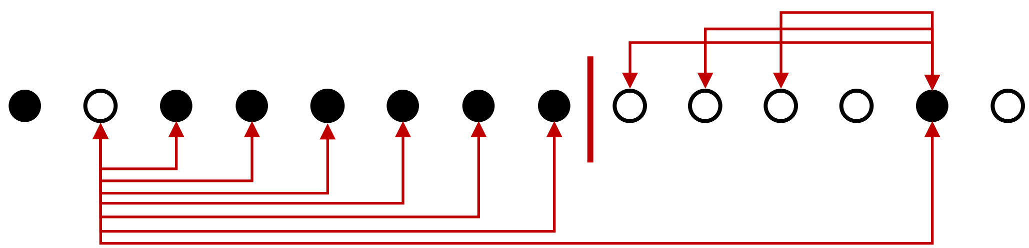
Absolute preferences given, so in principle an ordinal regression problem. However, the goal is to maximize ranking instead of classification performance.

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 (kernelized) ranking functions:

convex upper bound on $\mathbb{I}\left(f(\boldsymbol{x}) < f(\boldsymbol{x}')\right)$

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

check for all positive/negative pairs

regularizer

→ the training set scales QUADRATICALLY with the number of data points!

RankSVM AND RELATED METHODS (BIPARTITE CASE)

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

regularizer

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

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 $\mathcal{X} \rightarrow \Omega$, 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 in appropriate 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

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

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
2. Performance Assessment and Loss Functions
3. **Preference Learning Techniques**
   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
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** 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:

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

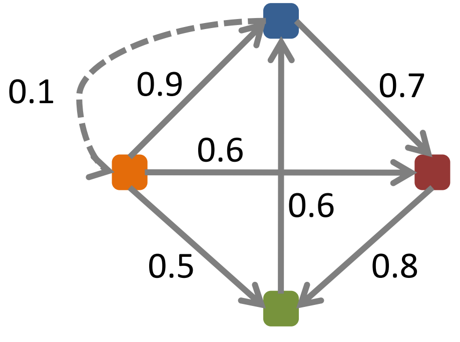
- The weights can be learned, for example, by means of the weighted majority algorithm [Littlestone & Warmuth 94].
- In a second step, a total order is derived, which is as much as possible in agreement with the binary preference relation.

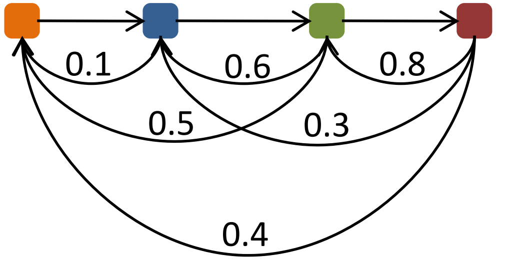
OBJECT RANKING: LEARNING TO ORDER THINGS [Cohen et al. 99]

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

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

  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.

LEARNING BY PAIRWISE COMPARISON (LPC) [Hüllermeier et al. 2008]

Label ranking is the problem of learning a function $\mathcal{X} \rightarrow \Omega$, 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

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

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**.

LEARNING BY PAIRWISE COMPARISON (LPC) [Hüllermeier et al. 2008]

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

X1	X2	X3			class
0.34	0	10			1
~~1.45~~	~~0~~	~~32~~			
1.22	1	46			0
0.74	1	25			1
~~0.95~~	~~1~~	~~72~~	~~273~~		
1.04	0	33	158	D ≻ A, **A ≻ B**, C ≻ B, A ≻ C	1

X1	X2	X3	X4	class
0.34	0	10	174	1
1.22	1	46	421	0
0.74	1	25	165	1
1.04	0	33	158	1

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	

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**.

DECOMPOSITION IN LEARNING RANKING FUNCTIONS

- A **ranking function** (mapping sets to permutations) is represented as
  - an **aggregation of individual utilitiy degrees** (argsort), or
  - as an **aggregation of pairwise preferences**.
- The corresponding **univariate** resp**. bivariate models** can be trained
  - **independently of each other**, or
  - **simultaneously** (in a coordinated manner).
- This also depends on the question whether the **target loss function** (defined on rankings) is decomposable, too.
- Information retrieval terminology:
  - **„pointwise learning“**: independent training of univariate models,
  - **„pairwise learning“**: independent training of bivariate models,
  - **„listwise learning“**: simultaneous learning of univariate models (direct minimization of a ranking loss)

AGENDA

1. Preference Learning Tasks
2. Performance Assessment and Loss Functions
3. **Preference Learning Techniques**
    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
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 ↓

s.t. $\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i, \boldsymbol{y}_i) \rangle - \langle \boldsymbol{w}, \phi(\boldsymbol{x}_i, \boldsymbol{y}) \rangle \geq \Delta(\boldsymbol{y}_i, \boldsymbol{y}) - \xi_i$ 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
2. Performance Assessment and Loss Functions
3. **Preference Learning Techniques**
   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
5. Conclusions

MODEL-BASED METHODS FOR RANKING

- By **model-based approaches** to ranking we subsume methods that
  - 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

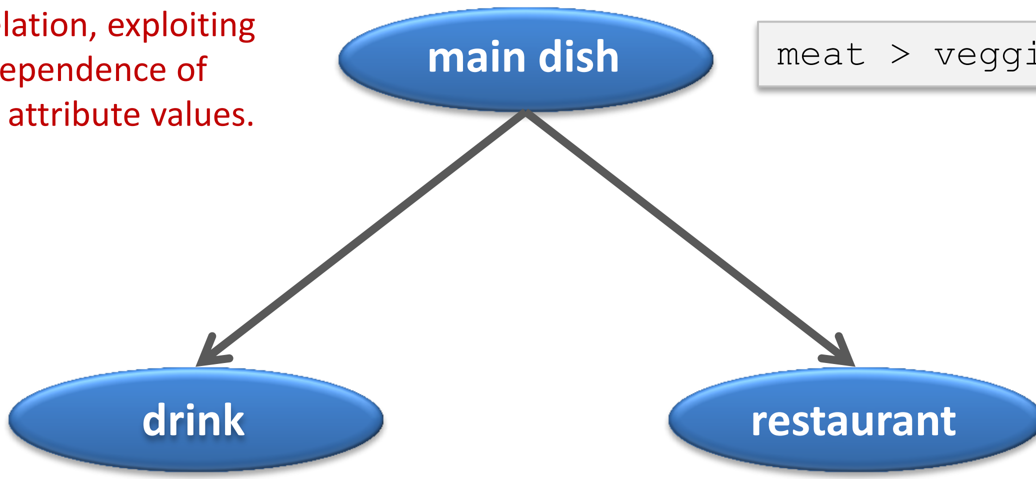
LEARNING LEXICOGRAPHIC PREFERENCE MODELS [Yaman et al. 2008]

- 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 NETWORKS [Chevaleyre et al. 2010]

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

```
meat > veggie > fish
```



label ranking problem

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

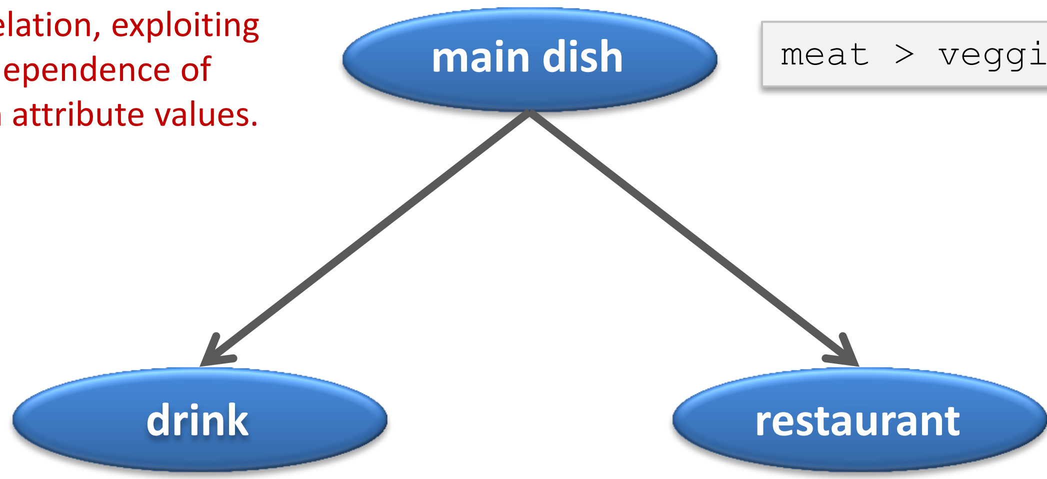
```
meat:   Italian > Chinese
veggie: Chinese > Italian
fish:   Chinese > Italian
```

Induces partial order relation, e.g.,

```
(meat, red wine, Italian)    >  (meat, white wine, Chinese)
(fish, white wine, Chinese) >  (fish, red wine, Chinese)
(meat, white wine, Italian) ?  (meat, red wine, Chinese)
```

LEARNING CONDITIONAL PREFERENCE 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:

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

PROBABILISTIC MODELS IN LABEL RANKING

input x $\mapsto$

permutation			probability
			0.2
			0
			0.1
			0.4
			0
			0.1

LABEL RANKING WITH 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

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

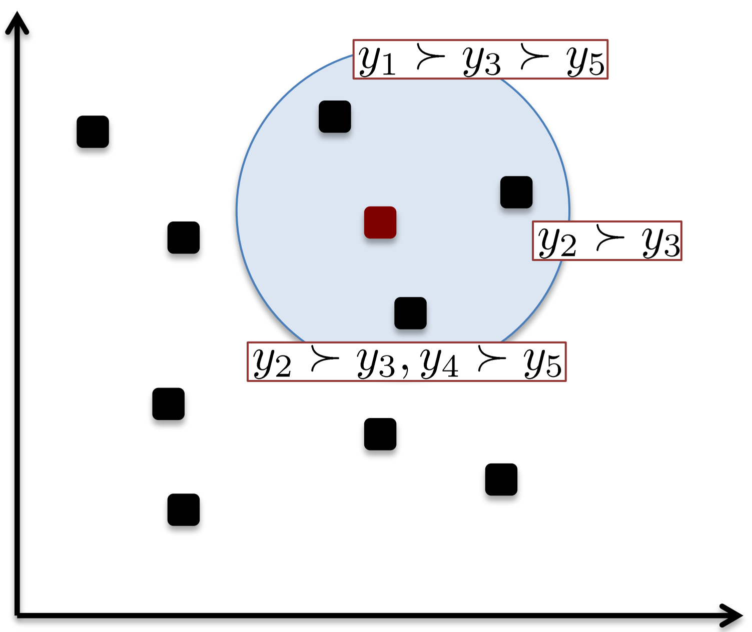
$$v(m,n) = \exp\left(\sum_{d=1}^{D} \alpha_d^{(m)} \cdot x_d^{(n)}\right) .$$

AGENDA

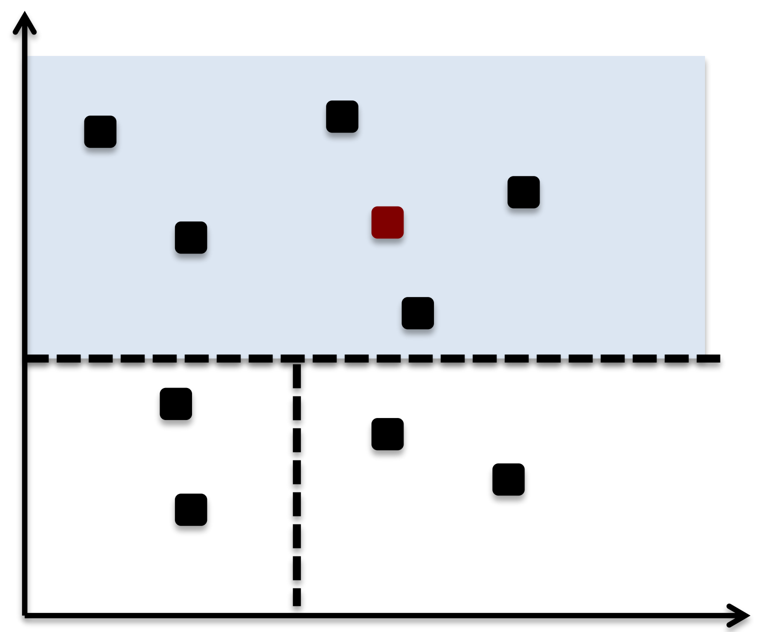
1. Preference Learning Tasks
2. Performance Assessment and Loss Functions
3. **Preference Learning Techniques**
   a. Learning Utility Functions
   b. Learning Preference Relations
   c. Structured Output Prediction
   d. Model-Based Preference Learning
   e. Local Preference Aggregation
4. Complexity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

LOCAL PREFERENCE AGGREGATION

- Estimation of a **piecewise constant** model (determining proper subregions of the instance space and considering observations therein as representative).



Nearest Neighbor Estimation



Decision Tree Learning

LOCAL PREFERENCE 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.

LOCAL PREFERENCE AGGREGATION

- 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.
- Has been done, for example, for the **Plackett-Luce model** and the **Mallows model**, both for complete rankings and pairwise comparisons[Cheng et al. 2009, 2010c].

Plackett-Luce

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

Mallows

$$\mathbf{P}(\pi \,|\, \pi_0, \theta) = \frac{\exp\big(-\theta\Delta(\pi, \pi_0)\big)}{\phi(\pi_0, \theta)}$$

ML ESTIMATION FOR THE MALLOWS MODEL [Cheng et al. 09]

set of (local) preferences

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

ML estimation →

center ranking

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

0.72

spread/precision

- Similar methods can also be used for other purposes, for example **clustering** using mixtures of probability distributions [Murphey & Martin 2003, Lu & Boutilier 2011].

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 **probabilistic ranking models** (e.g., Mallows, Plackett-Luce)
- **Restricted model classes**, fitting parametrized models such as lexicographic orders or CP nets.
- **Local preference aggregation** (lazy learning, recursive partitioning)

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.
- T. Lu and C. Boutilier. Learning Mallows models with pairwise preferences. ICML 2011.
- T.B. Murphey and D. Martin. Mixtures of distance-based models for ranking data. Comp. Statistics and Data Analysis, 41, 645-655, 2003.
- 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.