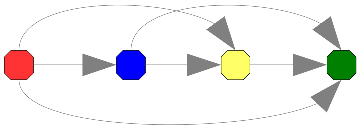
AGENDA

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

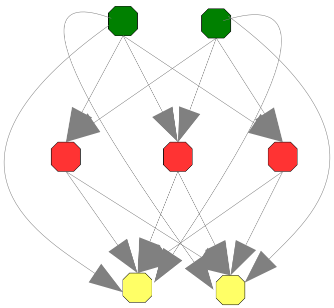
Training Complexity: Number of Preferences

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

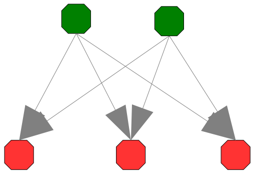
- total ranking: $d = \frac{c \cdot (c-1)}{2}$



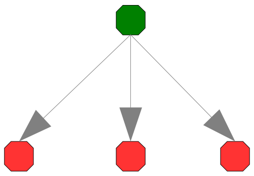
- multi-partite ranking *(k partitions with p_i items each):* $d = \sum_{i \neq j} p_i \cdot p_j$



- 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 $\sum_{ij} n_{ij} = d$

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

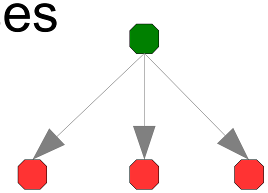
[Hüllermeier et al. 2008]

- 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: $c \cdot \log(c)$

$$\boxed{O(c \cdot (\log(c) + f))}$$

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

$$\boxed{O(c^2 \cdot f)}$$

  - 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)

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

  - 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 $c_j \in K \setminus \{c_a\}$ 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

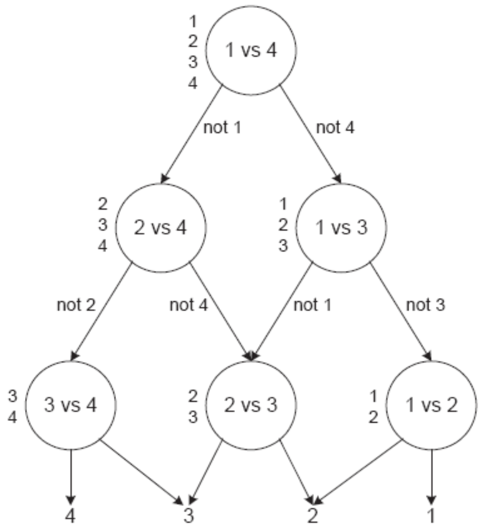
evaluate the classifier and update loss statistics

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

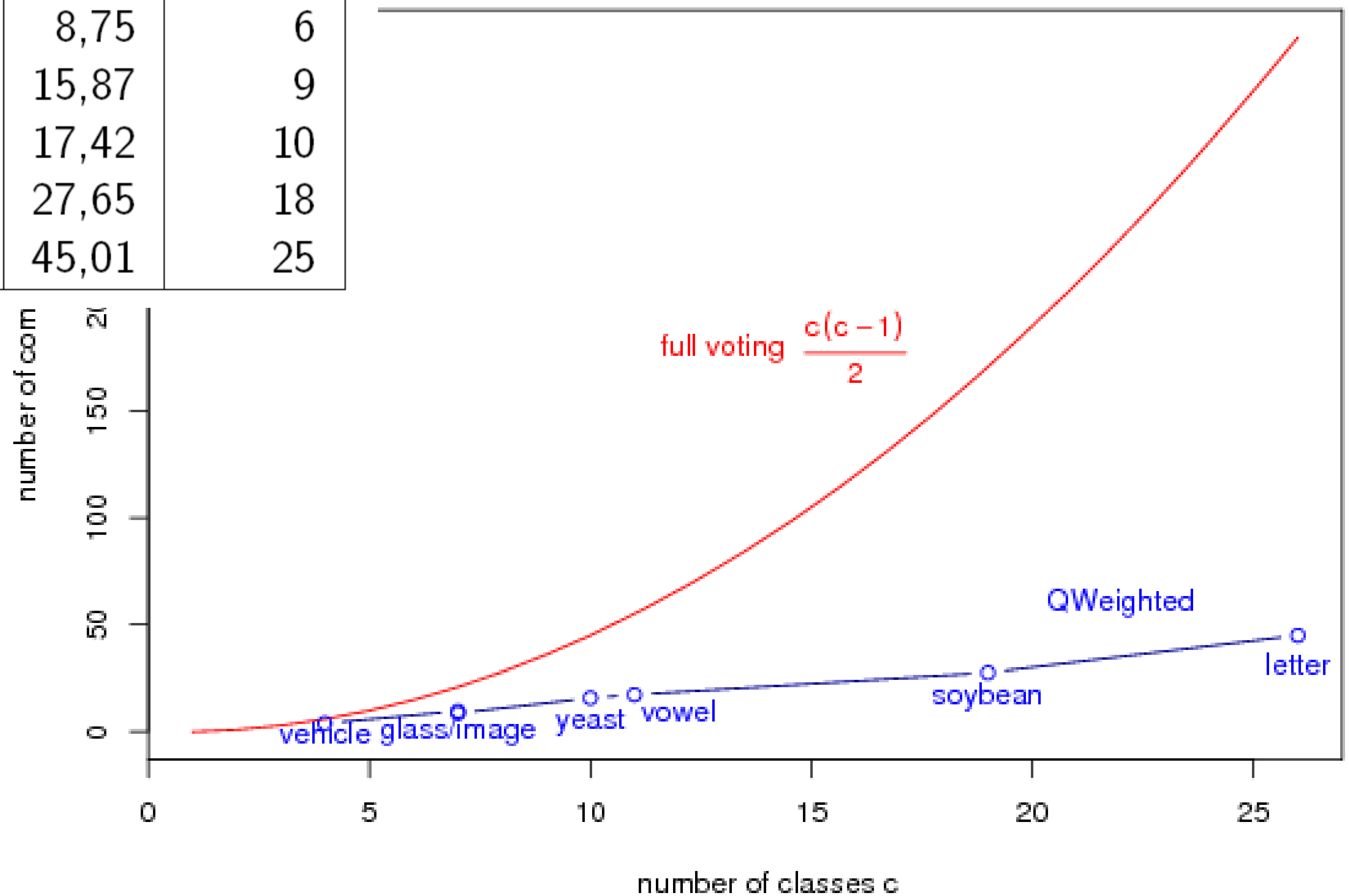
- DDAGs slightly faster
- but considerably less accurate

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