Artificial Intelligence 172 (2008) 1897–1916

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Label ranking by learning pairwise preferences

Eyke Hüllermeier[a,*], Johannes Fürnkranz[b], Weiwei Cheng[a], Klaus Brinker[a]

[a] *Department of Mathematics and Computer Science, Philipps-Universität Marburg, Germany*
[b] *Department of Computer Science, TU Darmstadt, Germany*

A R T I C L E   I N F O

Article history:
Received 21 January 2008
Received in revised form 14 July 2008
Accepted 8 August 2008
Available online 15 August 2008

Keywords:
Preference learning
Ranking
Pairwise classification
Constraint classification

A B S T R A C T

Preference learning is an emerging topic that appears in different guises in the recent literature. This work focuses on a particular learning scenario called label ranking, where the problem is to learn a mapping from instances to rankings over a finite number of labels. Our approach for learning such a mapping, called ranking by pairwise comparison (RPC), first induces a binary preference relation from suitable training data using a natural extension of pairwise classification. A ranking is then derived from the preference relation thus obtained by means of a ranking procedure, whereby different ranking methods can be used for minimizing different loss functions. In particular, we show that a simple (weighted) voting strategy minimizes risk with respect to the well-known Spearman rank correlation. We compare RPC to existing label ranking methods, which are based on scoring individual labels instead of comparing pairs of labels. Both empirically and theoretically, it is shown that RPC is superior in terms of computational efficiency, and at least competitive in terms of accuracy.

1. Introduction

The topic of *preferences* has recently attracted considerable attention in Artificial Intelligence (AI) research, notably in fields such as agents, non-monotonic reasoning, constraint satisfaction, planning, and qualitative decision theory [19].[1] Preferences provide a means for specifying desires in a declarative way, which is a point of critical importance for AI. In fact, consider AI's paradigm of a rationally acting (decision-theoretic) agent: The behavior of such an agent has to be driven by an underlying preference model, and an agent recommending decisions or acting on behalf of a user should clearly reflect that user's preferences.

It is hence hardly surprising that methods for learning and predicting preferences in an automatic way are among the very recent research topics in disciplines such as machine learning, knowledge discovery, and recommender systems. Many approaches have been subsumed under the terms of ranking and preference learning, even though some of them are quite different and are not sufficiently well discriminated by existing terminology. We will thus start our paper with a clarification of its contribution (Section 2). The learning scenario that we will consider in this paper assumes a collection of training examples which are associated with a finite set of decision alternatives. Following the common notation of supervised learning, we shall refer to the latter as *labels*. However, contrary to standard classification, a training example is not assigned a single label, but a set of *pairwise preferences* between labels (which neither has to be complete nor entirely

* Corresponding author.
E-mail addresses: eyke@informatik.uni-marburg.de (E. Hüllermeier), juffi@ke.informatik.tu-darmstadt.de (J. Fürnkranz), cheng@informatik.uni-marburg.de (W. Cheng), brinker@informatik.uni-marburg.de (K. Brinker).

[1] The increasing activity in this area is also witnessed by several workshops that have been devoted to preference learning and related topics, such as those at the NIPS-02, KI-03, SIGIR-03, NIPS-04, GfKl-05, IJCAI-05 and ECAI-2006 conferences (the second and fifth organized by two of the authors).

0004-3702/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.artint.2008.08.002

Table 1
Four different approaches to learning from preference information together with representative references

	modeling utility functions	modeling pairwise preferences
object ranking	comparison training [58]	learning to order things [13]
label ranking	constraint classification [28]	**this work** [24]

consistent), each one expressing that one label is preferred over another. The goal is to learn to predict a total order, a *ranking*, of all possible labels for a new training example.

The *ranking by pairwise comparison* (RPC) algorithm, which we introduce in Section 3 of this paper, has a modular structure and works in two phases. First, pairwise preferences are learned from suitable training data, using a natural extension of so-called *pairwise classification*. Then, a ranking is derived from a set of such preferences by means of a *ranking procedure*. In Section 4, we analyze the computational complexity of the RPC algorithm. Then, in Section 5, it will be shown that, by using suitable ranking procedures, RPC can minimize the risk for certain loss functions on rankings. Section 6 is devoted to an experimental evaluation of RPC and a comparison with alternative approaches applicable to the same learning problem. The paper closes with a discussion of related work in Section 7 and concluding remarks in Section 8. Parts of this paper are based on [24,25,33].

2. Learning from preferences

In this section, we will motivate preference learning[2] as a theoretically interesting and practically relevant subfield of machine learning. One can distinguish two types of preference learning problems, namely *learning from object preferences* and *learning from label preferences*, as well as two different approaches for modeling the preferences, namely by *evaluating* individual alternatives (by means of a utility function), or by *comparing* (pairs of) competing alternatives (by means of a preference relation). Table 1 shows the four possible combinations thus obtained. In this section, we shall discuss these options and show that our approach, label ranking by pairwise comparison, is still missing in the literature and hence a novel contribution.

2.1. Learning from object preferences

The most frequently studied problem in learning from preferences is to induce a *ranking function* $r(\cdot)$ that is able to order any subset $\mathcal{O}$ of an underlying class $\mathcal{X}$ of objects. That is, $r(\cdot)$ assumes as input a subset $\mathcal{O} = \{x_1 \dots x_n\} \subseteq \mathcal{X}$ of objects and returns as output a permutation τ of $\{1 \dots n\}$. The interpretation of this permutation is that object x_i is preferred to x_j whenever $\tau(i) < \tau(j)$. The objects themselves (e.g. websites) are typically characterized by a finite set of features as in conventional attribute-value learning. The training data consists of a set of exemplary pairwise preferences. This scenario, summarized in Fig. 1, is also known as “learning to order things” [13].

As an example consider the problem of learning to rank query results of a search engine [35,52]. The training information is provided implicitly by the user who clicks on some of the links in the query result and not on others. This information can be turned into binary preferences by assuming that the selected pages are preferred over nearby pages that are not clicked on [36].

2.2. Learning from label preferences

In this learning scenario, the problem is to predict, for any instance x (e.g., a person) from an instance space $\mathcal{X}$, a preference relation $\succ_x \subseteq \mathcal{L} \times \mathcal{L}$ among a finite set $\mathcal{L} = \{\lambda_1 \dots \lambda_m\}$ of labels or alternatives, where $\lambda_i \succ_x \lambda_j$ means that instance x prefers the label λ_i to the label λ_j. More specifically, we are especially interested in the case where $\succ_x$ is a total strict order, that is, a ranking of $\mathcal{L}$. Note that a ranking $\succ_x$ can be identified with a permutation τ_x of $\{1 \dots m\}$, e.g., the permutation τ_x such that $\tau_x(i) < \tau_x(j)$ whenever $\lambda_i \succ_x \lambda_j$ ($\tau(i)$ is the position of λ_i in the ranking). We shall denote the class of all permutations of $\{1 \dots m\}$ by $\mathcal{S}_m$. Moreover, by abuse of notation, we shall sometimes employ the terms “ranking” and “permutation” synonymously.

The training information consists of a set of instances for which (partial) knowledge about the associated preference relation is available (cf. Fig. 2). More precisely, each training instance x is associated with a subset of all pairwise preferences. Thus, even though we assume the existence of an underlying (“true”) ranking, we do not expect the training data to provide full information about that ranking. Besides, in order to increase the practical usefulness of the approach, we even allow for inconsistencies, such as pairwise preferences which are conflicting due to observation errors.

[2] We interpret the term “preference” not literally but in a wide sense as a kind of order relation. Thus, $a \succ b$ can indeed mean that alternative a is more liked by a person than b, but also that a is an algorithm that outperforms b on a certain problem, that a is an event that is more probable than b, that a is a student finishing her studies before b, etc.

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

Find:
- a ranking function $r(\cdot)$ that assumes as input a set of objects $\mathcal{O} \subseteq \mathcal{X}$ and returns a permutation (ranking) of this set

Fig. 1. Learning from object preferences.

Given:
- a set of training instances $\{x_k \mid k = 1 \ldots n\} \subseteq \mathcal{X}$
  (each instance typically represented by a feature vector)
- a set of labels $\mathcal{L} = \{\lambda_i \mid i = 1 \ldots m\}$
- for each training instance x_k: a set of *pairwise preferences* of the form $\lambda_i \succ_{x_k} \lambda_j$

Find:
- a ranking function that maps any $x \in \mathcal{X}$ to a ranking $\succ_x$ of $\mathcal{L}$ (permutation $\tau_x \in \mathcal{S}_m$)

Fig. 2. Learning from label preferences.

As in the case of object ranking, this learning scenario has a large number of practical applications. In the empirical part, we investigate the task of predicting a "qualitative" representation of a gene expression profile as measured by microarray analysis from phylogenetic profile features [4]. Another application scenario is meta-learning, where the task is to rank learning algorithms according to their suitability for a new dataset, based on the characteristics of this dataset [10]. Finally, every preference statement in the well-known CP-nets approach [7], a qualitative graphical representation that reflects conditional dependence and independence of preferences under a *ceteris paribus* interpretation, formally corresponds to a label ranking.

In addition, it has been observed by several authors [17,24,28] that many conventional learning problems, such as classification and multi-label classification, may be formulated in terms of label preferences:

- *Classification:* A single class label λ_i is assigned to each example x_k. This implicitly defines the set of preferences $\{\lambda_i \succ_{x_k} \lambda_j \mid 1 \leqslant j \neq i \leqslant m\}$.
- *Multi-label classification:* Each training example x_k is associated with a subset $L_k \subseteq \mathcal{L}$ of possible labels. This implicitly defines the set of preferences $\{\lambda_i \succ_{x_k} \lambda_j \mid \lambda_i \in L_k, \lambda_j \in \mathcal{L} \setminus L_k\}$.

In each of the former scenarios, a ranking model $f : \mathcal{X} \to \mathcal{S}_m$ is learned from a subset of all possible pairwise preferences. A suitable projection may be applied to the ranking model (which outputs permutations) as a post-processing step, for example a projection to the top-rank in classification learning where only this label is relevant.

2.3. Learning utility functions

As mentioned above, one natural way to represent preferences is to evaluate individual alternatives by means of a (real-valued) utility function. In the object preferences scenario, such a function is a mapping $f : \mathcal{X} \to \mathbb{R}$ that assigns a utility degree $f(x)$ to each object x and, hence, induces a complete order on $\mathcal{X}$. In the label preferences scenario, a utility function $f_i : \mathcal{X} \to \mathbb{R}$ is needed for each of the labels λ_i, $i = 1 \ldots m$. Here, $f_i(x)$ is the utility assigned to alternative λ_i by instance x. To obtain a ranking for x, the alternatives are ordered according to these utility scores, i.e., $\lambda_i \succeq_x \lambda_j \Leftrightarrow f_i(x) \geqslant f_j(x)$.

If the training data would offer the utility scores directly, preference learning would reduce to a standard regression problem (up to a monotonic transformation of the utility values). This information can rarely be assumed, however. Instead, usually only constraints derived from comparative preference information of the form "This object (or label) should have a higher utility score than that object (or label)" are given. Thus, the challenge for the learner is to find a function that is as much as possible in agreement with all constraints.

For object ranking approaches, this idea has first been formalized by Tesauro [58] under the name *comparison training*. He proposed a symmetric neural-network architecture that can be trained with representations of two states and a training signal that indicates which of the two states is preferable. The elegance of this approach comes from the property that one can replace the two symmetric components of the network with a single network, which can subsequently provide a real-valued evaluation of single states. Later works on learning utility function from object preference data include [27,31, 35,60]

Subsequently, we outline two approaches, constraint classification (CC) and log-linear models for label ranking (LL), that are direct alternatives to our method of ranking by pairwise comparison, and that we shall later on compare with.

2.3.1. *Constraint classification*

For the case of label ranking, a corresponding method for learning the functions $f_i(\cdot)$, $i = 1 \ldots m$, from training data has been proposed in the framework of *constraint classification* [28,29]. Proceeding from linear utility functions

$$f_i(x) = \sum_{k=1}^{n} \alpha_{ik} x_k \tag{2.1}$$

with label-specific coefficients α_{ik}, $k = 1 \ldots n$, a preference $\lambda_i \succ_x \lambda_j$ translates into the constraint $f_i(x) - f_j(x) > 0$ or, equivalently, $f_j(x) - f_i(x) < 0$. Both constraints, the positive and the negative one, can be expressed in terms of the sign of an inner product $\langle z, \alpha \rangle$, where $\alpha = (\alpha_{11} \ldots \alpha_{1n}, \alpha_{21} \ldots \alpha_{mn})$ is a concatenation of all label-specific coefficients. Correspondingly, the vector z is constructed by mapping the original ℓ-dimensional training example $x = (x_1 \ldots x_\ell)$ into an $(m \times \ell)$-dimensional space: For the positive constraint, x is copied into the components $((i-1) \times \ell + 1) \ldots (i \times \ell)$ and its negation $-x$ into the components $((j-1) \times \ell + 1) \ldots (j \times \ell)$; the remaining entries are filled with 0. For the negative constraint, a vector is constructed with the same elements but reversed signs. Both constraints can be considered as training examples for a conventional binary classifier in an $(m \times \ell)$-dimensional space: The first vector is a positive and the second one a negative example. The corresponding learner tries to find a separating hyperplane in this space, that is, a suitable vector α satisfying all constraints. For classifying a new example e, the labels are ordered according to the response resulting from multiplying e with the i-th ℓ-element section of the hyperplane vector. To work with more general types of utility functions, the method can obviously be kernelized.

Alternatively, Har-Peled et al. [28,29] propose an online version of constraint classification, namely an iterative algorithm that maintains weight vectors $\alpha_1 \ldots \alpha_m \in \mathbb{R}^\ell$ for each label individually. In every iteration, the algorithm checks each constraint $\lambda_i \succ_x \lambda_j$ and, in case the associated inequality $\alpha_i \times x = f_i(x) > f_j(x) = \alpha_j \times x$ is violated, adapts the weight vectors α_i, α_j appropriately. In particular, using perceptron training, the algorithm can be implemented in terms of a multi-output perceptron in a way quite similar to the approach of Grammer and Singer [15].

2.3.2. *Log-linear models for label ranking*

So-called log-linear models for label ranking have been proposed in Dekel et al. [17]. Here, utility functions are expressed in terms of linear combinations of a set of *base ranking functions*:

$$f_i(x) = \sum_j \alpha_j h_j(x, \lambda_i),$$

where a base function $h_j(\cdot)$ maps instance/label pairs to real numbers. Interestingly, for the special case in which instances are represented as feature vectors $x = (x_1 \ldots x_\ell)$ and the base functions are of the form

$$h_{kj}(x, \lambda) = \begin{cases} x_k & \lambda = \lambda_j \\ 0 & \lambda \neq \lambda_j \end{cases} \quad (1 \leqslant k \leqslant \ell,\ 1 \leqslant j \leqslant m), \tag{2.2}$$

the approach is essentially equivalent to CC, as it amounts to learning class-specific utility functions (2.1). Algorithmically, however, the underlying optimization problem is approached in a different way, namely by means of a boosting-based algorithm that seeks to minimize a (generalized) ranking error in an iterative way.

2.4. *Learning preference relations*

The key idea of this approach is to model the individual preferences directly instead of translating them into a utility function. This seems a natural approach, since it has already been noted that utility scores are difficult to elicit and observed preferences are usually of the relational type. For example, it is very hard to ensure a consistent scale even if all utility evaluations are performed by the same user. The situation becomes even more problematic if utility scores are elicited from different users, which may not have a uniform scale of their scores [13]. For the learning of preferences, one may bring up a similar argument. It will typically be easier to learn a separate theory for each individual preference that compares two objects or two labels and determines which one is better. Of course, every learned utility function that assigns a score to a set of labels $\mathcal{L}$ induces such a binary preference relation on these labels.

For object ranking problems, the pairwise approach has been pursued in [13]. The authors propose to solve object ranking problems by learning a binary preference predicate $Q(x, x')$, which predicts whether x is preferred to x' or vice versa. A final ordering is found in a second phase by deriving a ranking that is maximally consistent with these predictions.

For label ranking problems, the pairwise approach has been introduced by Fürnkranz and Hüllermeier [24]. The key idea, to be described in more detail in Section 3, is to learn, for each pair of labels (λ_i, λ_j), a binary predicate $\mathcal{M}_{ij}(x)$ that predicts whether $\lambda_i \succ_x \lambda_j$ or $\lambda_j \succ_x \lambda_i$ for an input x. In order to rank the labels for a new object, predictions for all pairwise label preferences are obtained and a ranking that is maximally consistent with these preferences is derived.

3. Label ranking by learning pairwise preferences

The key idea of *ranking by pairwise comparison* (RPC) is to reduce the problem of label ranking to several binary classification problems (Sections 3.1 and 3.2). The predictions of this ensemble of binary classifiers can then be combined into a ranking using a separate ranking algorithm (Section 3.3). We consider this *modularity* of RPC as an important advantage of the approach. Firstly, the binary classification problems are comparably simple and efficiently learnable. Secondly, as will become clear in the remainder of the paper, different ranking algorithms allow the ensemble of pairwise classifiers to adapt to different loss functions on label rankings without the need for re-training the pairwise classifiers.

3.1. Pairwise classification

The key idea of pairwise learning is well-known in the context of classification [22], where it allows one to transform a multi-class classification problem, i.e., a problem involving $m > 2$ classes $\mathcal{L} = \{\lambda_1 \dots \lambda_m\}$, into a number of *binary* problems. To this end, a separate model (base learner) $\mathcal{M}_{ij}$ is trained for each *pair* of labels $(\lambda_i, \lambda_j) \in \mathcal{L}$, $1 \leqslant i < j \leqslant m$; thus, a total number of $m(m-1)/2$ models is needed. $\mathcal{M}_{ij}$ is intended to separate the objects with label λ_i from those having label λ_j. At classification time, a query instance is submitted to all models $\mathcal{M}_{ij}$, and their predictions are combined into an overall prediction. In the simplest case, each prediction of a model $\mathcal{M}_{ij}$ is interpreted as a vote for either λ_i or λ_j, and the label with the highest number of votes is proposed as a final prediction.[3]

Pairwise classification has been tried in the areas of statistics [8,21], neural networks [40,41,44,51], support vector machines [30,32,42,54], and others. Typically, the technique learns more accurate theories than the more commonly used one-against-all classification method, which learns one theory for each class, using the examples of this class as positive examples and all others as negative examples.[4] Surprisingly, it can be shown that pairwise classification is also computationally more efficient than one-against-all class binarization (cf. Section 4).

3.2. Learning pairwise preference

The above procedure can be extended to the case of preference learning in a natural way [24]. Again, a preference (order) information of the form $\lambda_a \succ_x \lambda_b$ is turned into a training example (x, y) for the learner $\mathcal{M}_{ij}$, where $i = \min(a, b)$ and $j = \max(a, b)$. Moreover, $y = 1$ if $a < b$ and $y = 0$ otherwise. Thus, $\mathcal{M}_{ij}$ is intended to learn the mapping that outputs 1 if $\lambda_i \succ_x \lambda_j$ and 0 if $\lambda_j \succ_x \lambda_i$:

$$x \mapsto \begin{cases} 1 & \text{if } \lambda_i \succ_x \lambda_j, \\ 0 & \text{if } \lambda_j \succ_x \lambda_i. \end{cases} \tag{3.1}$$

The model is trained with all examples x_k for which either $\lambda_i \succ_{x_k} \lambda_j$ or $\lambda_j \succ_{x_k} \lambda_i$ is known. Examples for which nothing is known about the preference between λ_i and λ_j are ignored.

The mapping (3.1) can be realized by any binary classifier. Alternatively, one may also employ base classifiers that map into the unit interval $[0, 1]$ instead of $\{0, 1\}$, and thereby assign a *valued preference relation* $\mathcal{R}_x$ to every (query) instance $x \in \mathcal{X}$:

$$\mathcal{R}_x(\lambda_i, \lambda_j) = \begin{cases} \mathcal{M}_{ij}(x) & \text{if } i < j, \\ 1 - \mathcal{M}_{ji}(x) & \text{if } i > j \end{cases} \tag{3.2}$$

for all $\lambda_i \neq \lambda_j \in \mathcal{L}$. The output of a $[0, 1]$-valued classifier can usually be interpreted as a probability or, more generally, a kind of confidence in the classification: the closer the output of $\mathcal{M}_{ij}$ to 1, the stronger the preference $\lambda_i \succ_x \lambda_j$ is supported.

Fig. 3 illustrates the entire process for a hypothetical dataset with eight examples that are described with three binary attributes (*A1, A2, A3*) and preferences among three labels (*a, b, c*). First, the original training set is transformed into three two-class training sets, one for each possible pair of labels, containing only those training examples for which the relation between these two labels is known. Then three binary models, $\mathcal{M}_{ab}$, $\mathcal{M}_{bc}$, and $\mathcal{M}_{ac}$ are trained. In our example, the result could be simple rules like the following:

$\mathcal{M}_{ab}$: $a > b$ **if** $A2 = 1$.

$\mathcal{M}_{bc}$: $b > c$ **if** $A3 = 1$.

$\mathcal{M}_{ac}$: $a > c$ **if** $A1 = 1 \vee A3 = 1$.

Given a new example with an unknown preference structure (shown in the bottom left of Fig. 3), the predictions of these models are then used to predict a ranking for this example. As we will see in the next section, this is not always as trivial as in this example.

[3] Ties can be broken in favor or prevalent classes, i.e., according to the class distribution in the classification setting.

[4] Rifkin and Klautau [53] have argued that, at least in the case of support vector machines, one-against-all can be as effective provided that the binary base classifiers are carefully tuned.

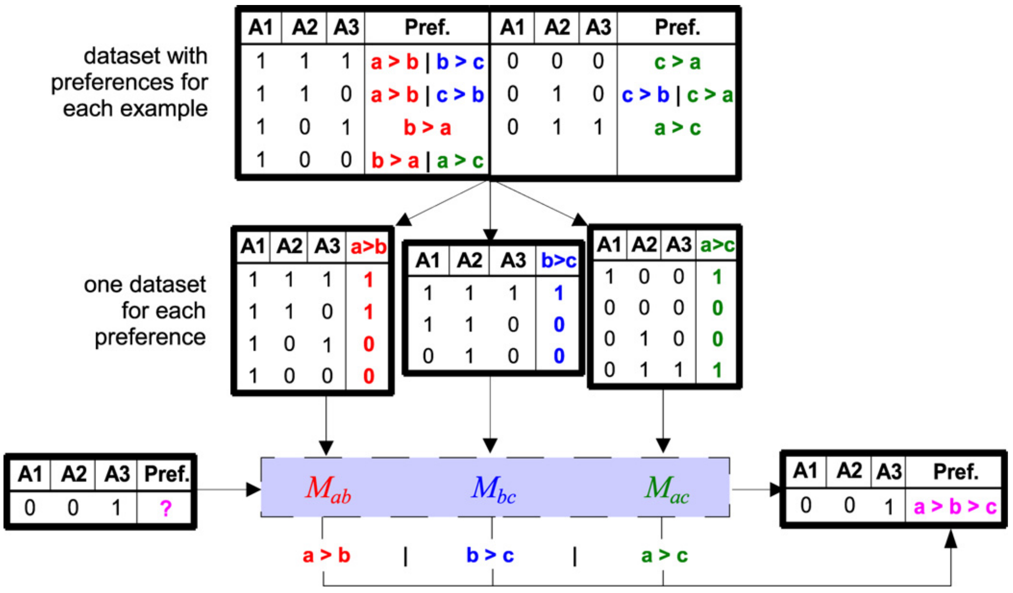

Fig. 3. Schematic illustration of learning by pairwise comparison.

3.3. *Combining predicted preferences into a ranking*

Given a predicted preference relation $\mathcal{R}_x$ for an instance x, the next question is how to derive an associated ranking τ_x. This question is non-trivial, since a relation $\mathcal{R}_x$ does not always suggest a unique ranking in an unequivocal way. For example, the learned preference relation need not be transitive (cf. Section 3.4). In fact, the problem of inducing a ranking from a (valued) preference relation has received a lot of attention in several research fields, e.g., in fuzzy preference modeling and (multi-attribute) decision making [20]. In the context of pairwise classification and preference learning, several studies have empirically compared different ways of combining the predictions of individual classifiers [2,23,34,62].

A simple though effective strategy is a generalization of the aforementioned voting strategy: each alternative λ_i is evaluated by the sum of (weighted) votes

$$S(\lambda_i) = \sum_{\lambda_j \neq \lambda_i} \mathcal{R}_x(\lambda_i, \lambda_j), \tag{3.3}$$

and all labels are then ordered according to these evaluations, i.e., such that

$$(\lambda_i \succ_x \lambda_j) \Rightarrow \big(S(\lambda_i) \geqslant S(\lambda_j)\big). \tag{3.4}$$

Even though this ranking procedure may appear rather ad-hoc at first sight, we shall give a theoretical justification in Section 5, where it will be shown that ordering the labels according to (3.3) minimizes a reasonable loss function on rankings.

3.4. *Transitivity*

Our pairwise learning scheme as outlined above produces a relation $\mathcal{R}_x$ by learning the preference degrees $\mathcal{R}_x(\lambda_i, \lambda_j)$ independently of each other. In this regard, one may wonder whether there are no interdependencies between these degrees that should be taken into account. In particular, as *transitivity* of pairwise preferences is one of the most important properties in preference modeling, an interesting question is whether any sort of transitivity can be guaranteed for $\mathcal{R}_x$.

Obviously, the learned binary preference relation does not necessarily have the typical properties of order relations. For example, transitivity will in general not hold, because if $\lambda_i \succ_x \lambda_j$ and $\lambda_j \succ_x \lambda_k$, the independently trained classifier $\mathcal{M}_{ik}$ may still predict $\lambda_k \succ_x \lambda_i$.[5] This is not a problem, because the subsequent ranking phase will convert the intransitive predictive preference relation into a total preference order.

However, it can be shown that, given the formal assumptions of our setting, the following weak form of transitivity must be satisfied:

$$\forall i, j, k \in \{1 \dots m\}\colon\ \mathcal{R}_x(\lambda_i, \lambda_j) \geqslant \mathcal{R}_x(\lambda_i, \lambda_k) + \mathcal{R}_x(\lambda_k, \lambda_j) - 1. \tag{3.5}$$

[5] In fact, not even symmetry needs to hold if $\mathcal{M}_{ij}$ and $\mathcal{M}_{ji}$ are different models, which is, e.g., the case for rule learning algorithms [22]. This situation may be compared with round robin sports tournament, where individual results do not necessarily conform to the final ranking that is computed from them.

As a consequence of this property, which is proved in Appendix A, the predictions obtained by an ensemble of pairwise learners $\mathcal{M}_{ij}$ should actually satisfy (3.5). In other words, training the learners independently of each other is indeed not fully legitimate. Fortunately, our experience so far has shown that the probability to violate (3.5) is not very high. Still, forcing (3.5) to hold is a potential point of improvement and part of ongoing work.

4. Complexity analysis

In this section, we will generalize previous results on the efficiency of pairwise classification to preference learning. In particular, we will show that this approach can be expected to be computationally more efficient than alternative approaches like constraint classification that try to model the preference learning problem as a single binary classification problem in a higher-dimensional space (cf. Section 2.3).

4.1. Ranking by pairwise comparison

First, we will bound the number of training examples used by the pairwise approach. Let $|P_k|$ be the number of preferences that are associated with example x_k. Throughout this section, we denote by $d = 1/n \cdot \sum_k |P_k|$ the average number of preferences over all examples.

Lemma 1. *The total number of training examples constructed by RPC is $n \cdot d$, which is bounded by $n \cdot m(m-1)/2$, i.e.,*

$$\sum_{k=1}^{n} |P_k| = n \cdot d \leqslant n \cdot \frac{m(m-1)}{2}.$$

Proof. Each of the n training examples will be added to all $|P_k|$ binary training sets that correspond to one of its preferences. Thus, the total number of training examples is $\sum_{k=1}^{n} |P_k| = n \cdot d$. This is bounded from above by the size of a complete set of preferences $n \cdot m(m-1)/2$. $\square$

The special case for classification, where the number of training examples grow only linearly with the number of classes [22], can be obtained as a corollary of this theorem, because for classification, each class label expands to $d = m - 1$ preferences.

As a consequence, it follows immediately that RPC using a base algorithm with a linear run-time complexity $\mathcal{O}(n)$ has a total run-time of $\mathcal{O}(d \cdot n)$. More interesting is the general case.

Theorem 1. *For a base learner with complexity $\mathcal{O}(n^a)$, the complexity of RPC is $\mathcal{O}(d \cdot n^a)$.*

Proof. Let n_{ij} be the number of training examples for model $\mathcal{M}_{ij}$. Each example corresponds to a single preference, i.e.,

$$\sum_{1 \leqslant i < j \leqslant m} n_{ij} = \sum_{k=1}^{n} |P_k| = d \cdot n$$

and the total learning complexity is $\sum \mathcal{O}(n_{ij}^a)$. We now obtain

$$\frac{\sum \mathcal{O}(n_{ij}^a)}{\mathcal{O}(d \cdot n^a)} = \frac{1}{d} \sum \frac{\mathcal{O}(n_{ij}^a)}{\mathcal{O}(n^a)} = \frac{1}{d} \sum \mathcal{O}\left(\left(\frac{n_{ij}}{n}\right)^a\right)$$
$$\leqslant \frac{1}{d} \sum \mathcal{O}\left(\frac{n_{ij}}{n}\right) = \frac{\sum \mathcal{O}(n_{ij})}{d \cdot \mathcal{O}(n)} = \frac{\mathcal{O}(\sum n_{ij})}{\mathcal{O}(d \cdot n)} = \frac{\mathcal{O}(d \cdot n)}{\mathcal{O}(d \cdot n)} = \mathcal{O}(1).$$

The inequality holds because each example can have at most one preference involving the pair of labels (λ_i, λ_j). Thus, $n_{ij} \leqslant n$. $\square$

Again, we obtain as a corollary that the complexity of pairwise classification is only linear in the number of classes $\mathcal{O}(m \cdot n^a)$, for which an incomplete proof was previously given in [22].

4.2. Constraint classification and log-linear models

For comparison, CC converts each example into a set of examples, one positive and one negative for each preference. This construction leads to the following complexity.

Theorem 2. *For a base learner with complexity $\mathcal{O}(n^a)$, the total complexity of constraint classification is $\mathcal{O}(d^a \cdot n^a)$.*

Proof. CC transforms the original training data into a set of $2\sum_{k=1}^{n}|P_k| = 2dn$ examples, which means that CC constructs twice as many training examples as RPC. If this problem is solved with a base learner with complexity $\mathcal{O}(n^a)$, the total complexity is $\mathcal{O}((2dn)^a) = \mathcal{O}(d^a \cdot n^a)$. $\square$

Moreover, the newly constructed examples are projected into a space that has m times as many attributes as the original space.

A direct comparison is less obvious for the online version of CC whose complexity strongly depends on the number of iterations needed to achieve convergence. In a single iteration, the algorithm checks all constraints for every instance and, in case a constraint is violated, adapts the weight vector correspondingly. The complexity is hence $\mathcal{O}(n \cdot d \cdot \ell \cdot T)$, where ℓ is the number of attributes of an instance (dimension of the instance space) and T the number of iterations.

For the same reason, it is difficult to compare RPC with the boosting-based algorithm proposed for log-linear models by Dekel et al. [17]. In each iteration, the algorithm essentially updates the weights that are associated with each instance and preference constraint. In the label ranking setting considered here, the complexity of this step is $\mathcal{O}(d \cdot n)$. Moreover, the algorithm maintains weight coefficients for each base ranking function. If specified as in (2.2), the number of these functions is $m \cdot \ell$. Therefore, the total complexity of LL is $\mathcal{O}((d \cdot n + m \cdot \ell) \cdot T)$, with T the number of iterations.

4.3. Discussion

In summary, the overall complexity of pairwise label ranking depends on the average number of preferences that are given for each training example. While being quadratic in the number of labels if a complete ranking is given, it is only linear for the classification setting. In any case, it is no more expensive than constraint classification and can be considerably cheaper if the complexity of the base learner is super-linear (i.e., $a > 1$). The comparison between RPC and LL is less obvious and essentially depends on how n^a relates to $n \cdot T$ (note that, implicitly, T also depends on n, as larger data sets typically need more iterations).

A possible disadvantage of RPC concerns the large number of classifiers that have to be stored. Assuming an input space $\mathcal{X}$ of dimensionality ℓ and simple linear classifiers as base learners, the pairwise approach has to store $\mathcal{O}(\ell \cdot m^2)$ parameters, whereas both CC and LL only need to store $\mathcal{O}(\ell \cdot m)$ parameters to represent their ranking model. (During training, however, the boosting-based optimization algorithm in LL must also store a typically much higher number of $n \cdot d$ parameters, one for each preference constraint.)

As all the model parameters have to be used for deriving a label ranking, this may also affect the prediction time. However, for the classification setting, it was shown in [48] that a more efficient algorithm yields the same predictions as voting in almost linear time ($\approx \mathcal{O}(\ell \cdot m)$). To what extent this algorithm can be generalized to label ranking is currently under investigation. As ranking is basically a sorting of all possible labels, we expect that this can be done in log-linear time ($\mathcal{O}(\ell \cdot m \log m)$).

5. Risk minimization

Even though the approach to pairwise ranking as outlined in Section 3 appears intuitively appealing, one might argue that it lacks a solid foundation and remains ad-hoc to some extent. For example, one might easily think of ranking procedures other than (3.3), leading to different predictions. In any case, one might wonder whether the rankings predicted on the basis of (3.2) and (3.3) do have any kind of optimality property. An affirmative answer to this question will be given in this section.

5.1. Preliminaries

Recall that, in the setting of label ranking, we associate every instance x from an instance space $\mathcal{X}$ with a ranking of a finite set of class labels $\mathcal{L} = \{\lambda_1 \ldots \lambda_m\}$ or, equivalently, with a permutation $\tau_x \in \mathcal{S}_m$ (where $\mathcal{S}_m$ denotes the class of all permutations of $\{1 \ldots m\}$). More specifically, and in analogy with the setting of conventional classification, every instance is associated with a *probability distribution* over the class of rankings (permutations) $\mathcal{S}_m$. That is, for every instance x, there exists a probability distribution $\mathbb{P}(\cdot \mid x)$ such that, for every $\tau \in \mathcal{S}_m$, $\mathbb{P}(\tau \mid x)$ is the probability to observe the ranking τ as an output, given the instance x as an input.

The quality of a model $\mathcal{M}$ (induced by a learning algorithm) is commonly measured in terms of its *expected loss* or *risk*

$$\mathbb{E}\big(D\big(y, \mathcal{M}(x)\big)\big), \tag{5.1}$$

where $D(\cdot)$ is a loss or distance function, $\mathcal{M}(x)$ denotes the prediction made by the model for the instance x, and y is the true outcome. The expectation $\mathbb{E}$ is taken over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{Y}$ is the output space;[6] in our case, $\mathcal{Y}$ is given by $\mathcal{S}_m$.

[6] The existence of a probability measure over $\mathcal{X} \times \mathcal{Y}$ must of course be assumed.

5.2. Spearman's rank correlation

An important and frequently applied similarity measure for rankings is the *Spearman rank correlation*, originally proposed by Spearman [57] as a nonparametric rank statistic to measure the strength of the associations between two variables [43]. It is defined as follows

$$1-\frac{6D(\tau,\tau')}{m(m^2-1)} \tag{5.2}$$

as a linear transformation (normalization) of the sum of squared rank distances

$$D(\tau',\tau)\overset{\mathrm{df}}{=}\sum_{i=1}^{m}\bigl(\tau'(i)-\tau(i)\bigr)^2 \tag{5.3}$$

to the interval $[-1,1]$. As will now be shown, RPC is a risk minimizer with respect to (5.3) (and hence Spearman rank correlation) as a distance measure under the condition that the binary models $\mathcal{M}_{ij}$ provide correct probability estimates, i.e.,

$$\mathcal{R}_x(\lambda_i,\lambda_j)=\mathcal{M}_{ij}(x)=\mathbb{P}(\lambda_i\succ_x\lambda_j). \tag{5.4}$$

That is, if (5.4) holds, then RPC yields a risk minimizing prediction

$$\hat{\tau}_x=\arg\min_{\tau\in\mathcal{S}_m}\sum_{\tau'\in\mathcal{S}_m}D(\tau,\tau')\cdot\mathbb{P}(\tau'\mid x) \tag{5.5}$$

if $D(\cdot)$ is given by (5.3). Admittedly, (5.4) is a relatively strong assumption, as it requires the pairwise preference probabilities to be perfectly learnable. Yet, the result (5.5) sheds light on the aggregation properties of our technique under ideal conditions and provides a valuable basis for further analysis. In fact, recalling that RPC consists of two steps, namely *pairwise learning* and *ranking*, it is clear that in order to study properties of the latter, some assumptions about the result of the former step have to be made. And even though (5.4) might at best hold approximately in practice, it seems to be at least as natural as any other assumption about the output of the ensemble of pairwise learners.

Lemma 2. *Let s_i, $i=1\dots m$, be real numbers such that $0\leqslant s_1\leqslant s_2\leqslant\cdots\leqslant s_m$. Then, for all permutations $\tau\in\mathcal{S}_m$,*

$$\sum_{i=1}^{m}(i-s_i)^2\leqslant\sum_{i=1}^{m}(i-s_{\tau(i)})^2. \tag{5.6}$$

Proof. We have

$$\begin{aligned}\sum_{i=1}^{m}(i-s_{\tau(i)})^2&=\sum_{i=1}^{m}(i-s_i+s_i-s_{\tau(i)})^2\\&=\sum_{i=1}^{m}(i-s_i)^2+2\sum_{i=1}^{m}(i-s_i)(s_i-s_{\tau(i)})+\sum_{i=1}^{m}(s_i-s_{\tau(i)})^2.\end{aligned}$$

Expanding the last equation and exploiting that $\sum_{i=1}^m s_i^2=\sum_{i=1}^m s_{\tau(i)}^2$ yields

$$\sum_{i=1}^{m}(i-s_{\tau(i)})^2=\sum_{i=1}^{m}(i-s_i)^2+2\sum_{i=1}^{m}i\,s_i-2\sum_{i=1}^{m}i\,s_{\tau(i)}.$$

On the right-hand side of the last equation, only the last term $\sum_{i=1}^m is_{\tau(i)}$ depends on τ. This term is maximal for $\tau(i)=i$, because $s_i\leqslant s_j$ for $i<j$, and therefore $\max_{i=1\dots m} ms_i=ms_m$, $\max_{i=1\dots m-1}(m-1)s_i=(m-1)s_{m-1}$, etc. Thus, the difference of the two sums is always positive, and the right-hand side is larger than or equal to $\sum_{i=1}^m(i-s_i)^2$, which proves the lemma. □

Lemma 3. *Let $\mathbb{P}(\cdot\mid x)$ be a probability distribution over $\mathcal{S}_m$. Moreover, let*

$$s_i\overset{\mathrm{df}}{=}m-\sum_{j\neq i}\mathbb{P}(\lambda_i\succ_x\lambda_j) \tag{5.7}$$

with

$$\mathbb{P}(\lambda_i\succ_x\lambda_j)=\sum_{\tau:\ \tau(i)<\tau(j)}\mathbb{P}(\tau\mid x). \tag{5.8}$$

Then, $s_i=\sum_\tau\mathbb{P}(\tau\mid x)\tau(i)$.

Proof. We have

$$\begin{aligned} s_i &= m - \sum_{j \neq i} \mathbb{P}(\lambda_i \succ_x \lambda_j) \\ &= 1 + \sum_{j \neq i} \bigl(1 - \mathbb{P}(\lambda_i \succ_x \lambda_j)\bigr) \\ &= 1 + \sum_{j \neq i} \mathbb{P}(\lambda_j \succ_x \lambda_i) \\ &= 1 + \sum_{j \neq i} \sum_{\tau:\ \tau(j) < \tau(i)} \mathbb{P}(\tau \mid x) \\ &= 1 + \sum_{\tau} \mathbb{P}(\tau \mid x) \sum_{j \neq i} \begin{cases} 1 & \text{if } \tau(i) > \tau(j) \\ 0 & \text{if } \tau(i) < \tau(j) \end{cases} \\ &= 1 + \sum_{\tau} \mathbb{P}(\tau \mid x)\bigl(\tau(i) - 1\bigr) \\ &= \sum_{\tau} \mathbb{P}(\tau \mid x)\tau(i). \qquad \square \end{aligned}$$

Note that $s_i \leqslant s_j$ is equivalent to $S(\lambda_i) \geqslant S(\lambda_j)$ (as defined in (3.3)) under the assumption (5.4). Thus, ranking the alternatives according to $S(\lambda_i)$ (in decreasing order) is equivalent to ranking them according to s_i (in increasing order).

Theorem 3. *The expected distance*

$$\mathbb{E}\bigl(D(\tau', \tau) \mid x\bigr) = \sum_{\tau} \mathbb{P}(\tau \mid x) \cdot D(\tau', \tau) = \sum_{\tau} \mathbb{P}(\tau \mid x) \sum_{i=1}^{m} \bigl(\tau'(i) - \tau(i)\bigr)^2$$

becomes minimal by choosing τ' such that $\tau'(i) \leqslant \tau'(j)$ whenever $s_i \leqslant s_j$, with s_i given by (5.7).

Proof. We have

$$\begin{aligned} \mathbb{E}\bigl(D(\tau', \tau) \mid x\bigr) &= \sum_{\tau} \mathbb{P}(\tau \mid x) \sum_{i=1}^{m} \bigl(\tau'(i) - \tau(i)\bigr)^2 \\ &= \sum_{i=1}^{m} \sum_{\tau} \mathbb{P}(\tau \mid x) \bigl(\tau'(i) - \tau(i)\bigr)^2 \\ &= \sum_{i=1}^{m} \sum_{\tau} \mathbb{P}(\tau \mid x) \bigl(\tau'(i) - s_i + s_i - \tau(i)\bigr)^2 \\ &= \sum_{i=1}^{m} \sum_{\tau} \mathbb{P}(\tau \mid x) \bigl[\bigl(\tau(i) - s_i\bigr)^2 - 2\bigl(\tau(i) - s_i\bigr)\bigl(s_i - \tau'(i)\bigr) + \bigl(s_i - \tau'(i)\bigr)^2\bigr] \\ &= \sum_{i=1}^{m} \Bigl[\sum_{\tau} \mathbb{P}(\tau \mid x)\bigl(\tau(i) - s_i\bigr)^2 - 2\bigl(s_i - \tau'(i)\bigr) \sum_{\tau} \mathbb{P}(\tau \mid x)\bigl(\tau(i) - s_i\bigr) + \sum_{\tau} \mathbb{P}(\tau \mid x)\bigl(s_i - \tau'(i)\bigr)^2\Bigr]. \end{aligned}$$

In the last equation, the mid-term on the right-hand side becomes 0 according to Lemma 3. Moreover, the last term obviously simplifies to $(s_i - \tau'(i))^2$, and the first term is a constant $c = \sum_{\tau} \mathbb{P}(\tau \mid x)(\tau(i) - s_i)^2$ that does not depend on τ'. Thus, we obtain $\mathbb{E}(D(\tau', \tau) \mid x) = c + \sum_{i=1}^{m} (s_i - \tau'(i))^2$ and the theorem follows from Lemma 2. $\square$

5.3. Kendall's tau

The above result shows that our approach to label ranking in the form presented in Section 3 is particularly tailored to (5.3) as a loss function. We like to point out, however, that RPC is not restricted to this measure but can also minimize other loss functions. As mentioned previously, this can be accomplished by replacing the ranking procedure in the second step of RPC in a suitable way. To illustrate, consider the well-known *Kendall tau measure* [38] as an alternative loss function. This measure essentially calculates the number of pairwise rank inversions on labels to measure the ordinal correlation of two rankings; more formally, with

$$D(\tau', \tau) \overset{\mathrm{df}}{=} \#\bigl\{(i, j) \bigm| i < j, \tau(i) > \tau(j) \wedge \tau'(i) < \tau'(j)\bigr\} \tag{5.9}$$

denoting the number of *discordant* pairs of items (labels), the Kendall tau coefficient is given by $1 - 4D(\tau', \tau)/(m(m-1))$, that is, by a linear scaling of $D(\tau', \tau)$ to the interval $[-1, +1]$.

Now, for every ranking τ',

$$\begin{aligned}
\mathbb{E}\big(D(\tau', \tau) \mid x\big) &= \sum_{\tau \in \mathcal{S}_m} \mathbb{P}(\tau) \times D(\tau', \tau) \\
&= \sum_{\tau \in \mathcal{S}_m} \mathbb{P}(\tau \mid x) \times \sum_{i<j \mid \tau'(i)<\tau'(j)} \begin{cases} 1 & \text{if } \tau(i) > \tau(j) \\ 0 & \text{if } \tau(i) < \tau(j) \end{cases} && (5.10) \\
&= \sum_{i<j \mid \tau'(i)<\tau'(j)} \sum_{\tau \in \mathcal{S}_m} \mathbb{P}(\tau \mid x) \times \begin{cases} 1 & \text{if } \tau(i) > \tau(j) \\ 0 & \text{if } \tau(i) < \tau(j) \end{cases} \\
&= \sum_{i<j \mid \tau'(i)<\tau'(j)} \mathbb{P}(\lambda_i \succ_x \lambda_j). && (5.11)
\end{aligned}$$

Thus, knowing the pairwise probabilities $\mathbb{P}(\lambda_i \succ_x \lambda_j)$ is again enough to derive the expected loss for every ranking τ'. In other words, RPC can also make predictions which are optimal for (5.9) as an underlying loss function. To this end, only the ranking procedure has to be adapted while the same pairwise probabilities (predictions of the pairwise learners) can be used.

Finding the ranking that minimizes (5.10) is formally equivalent to solving the graph-theoretical *feedback arc set* problem (for *weighted* tournaments) which is known to be NP complete [3]. Of course, in the context of label ranking, this result should be put into perspective, because the set of class labels is typically of small to moderate size. Nevertheless, from a computational point of view, the ranking procedure that minimizes Kendall's tau is definitely more complex than the procedure for minimizing Spearman's rank correlation.

5.4. Connections with voting theory

It is worth mentioning that the voting strategy in RPC, as discussed in Section 5.2, is closely related to the so-called *Borda-count*, a voting rule that is well-known in social choice theory [9]: Suppose that the preferences of n voters are expressed in terms of rankings $\tau_1, \tau_2 \ldots \tau_n$ of m alternatives. From a ranking τ_i, the following scores are derived for the alternatives: The best alternative receives $m-1$ points, the second best $m-2$ points, and so on. The overall score of an alternative is the sum of points that it has received from all voters, and a representative ranking $\hat{\tau}$ (aggregation of the single voters' rankings) is obtained by ordering the alternatives according to these scores.

Now, it is readily verified that the result obtained by this procedure corresponds exactly to the result of RPC if the probability distribution over the class $\mathcal{S}_m$ of rankings is defined by the corresponding relative frequencies. In other words, the ranking $\hat{\tau}$ obtained by RPC minimizes the sum of all distances:

$$\hat{\tau} = \arg\min_{\tau \in \mathcal{S}_m} \sum_{i=1}^{n} D(\tau, \tau_i). \tag{5.12}$$

A ranking of that kind is sometimes called *central ranking*.[7]

In connection with social choice theory it is also interesting to note that RPC does not satisfy the so-called *Condorcet criterion*: As the pairwise preferences in our above example show, it is thoroughly possible that an alternative (in this case λ_1) is preferred in all *pairwise* comparisons ($\mathcal{R}(\lambda_1, \lambda_2) > .5$ and $\mathcal{R}(\lambda_1, \lambda_3) > .5$) without being the overall winner of the election (top-label in the ranking). Of course, this apparently paradoxical property is not only relevant for ranking but also for classification. In this context, it has already been recognized by Hastie and Tibshirani [30].

Another distance (similarity) measure for rankings, which plays an important role in voting theory, is the aforementioned *Kendall tau*. When using the number of discordant pairs (5.9) as a distance measure $D(\cdot)$ in (5.12), $\hat{\tau}$ is also called the *Kemeny-optimal* ranking. Kendall's tau is intuitively quite appealing and Kemeny-optimal rankings have several nice properties. However, as noted earlier, one drawback of using Kendall's tau instead of rank correlation as a distance measure in (5.12) is a loss of computational efficiency. In fact, the computation of Kemeny-optimal rankings is known to be NP-hard [5].

6. Empirical evaluation

The experimental evaluation presented in this section compares, in terms of accuracy and computational efficiency, ranking by pairwise comparison (RPC) with weighted voting to the constraint classification (CC) approach and log-linear models for label ranking (LL) as outlined, respectively, in Sections 2.3.1 and 2.3.2. CC in particular is a natural counterpart

[7] See, e.g., Marden's book [45], which also contains results closely related to our results from Section 5.2.

Table 2
Statistics for the semi-synthetic and real datasets

dataset	#examples	#classes	#features
iris	150	3	4
wine	178	3	13
glass	214	6	9
vowel	528	11	10
vehicle	846	4	18
spo	2465	11	24
heat	2465	6	24
dtt	2465	4	24
cold	2465	4	24
diau	2465	7	24

to compare with, as its approach is orthogonal to ours: instead of breaking up the label ranking problem into a set of small pairwise learning problems, as we do, CC embeds the original problem into a single learning problem in a high-dimensional feature space. We implemented CC with support vector machines using a linear kernel as a binary classifier (CC-SVM).[8] Apart from CC in its original version, we also included an online-variant (CC-P) as proposed in [28], using a noise-tolerant perceptron algorithm as a base learner [37].[9]

To guarantee a fair comparison, we use LL with (2.2) as base ranking functions, which means that it is based on the same underlying model class as CC. Moreover, we implement RPC with simple logistic regression as a base learner,[10] which comes down to fitting a linear model and using the logistic link function ($\text{logit}(\pi) = \log(\pi/(1-\pi))$) to derive $[0,1]$-valued scores, the type of model output requested in RPC. Essentially, all three approaches are therefore based on linear models and, in fact, they all produce linear decision boundaries between classes.[11] Nevertheless, to guarantee full comparability between RPC and CC, we also implemented the latter with logistic regression as a base learner (CC-LR).

6.1. Datasets

To provide a comprehensive analysis under varying conditions, we considered different scenarios that can be roughly categorized as real-world and semi-synthetic.

The real-world scenario originates from the bioinformatics fields where ranking and multilabeled data, respectively, can frequently be found. More precisely, our experiments considered two types of genetic data, namely phylogenetic profiles and DNA microarray expression data for the Yeast genome.[12] The genome consists of 2465 genes, and each gene is represented by an associated phylogenetic profile of length 24. Using these profiles as input features, we investigated the task of predicting a "qualitative" representation of an expression profile: Actually, the expression profile of a gene is an ordered sequence of real-valued measurements, such as $(2.1, 3.5, 0.7, -2.5)$, where each value represents the expression level of that gene measured at a particular point of time. A qualitative representation can be obtained by converting the expression levels into ranks, i.e., ordering the time points (= labels) according to the associated expression values. In the above example, the qualitative profile would be given by $(2, 1, 3, 4)$, which means that the highest expression was observed at time point 2, the second-highest at time point 1, and so on. The use of qualitative profiles of that kind, and the Spearman correlation as a similarity measure between them, was motivated in [4], both biologically and from a data analysis point of view.

We used data from five microarray experiments (spo, heat, dtt, cold, diau), giving rise to five prediction problems all using the same input features but different target rankings. It is worth mentioning that these experiments involve different numbers of measurements, ranging from 4 to 11; see Table 2.[13] Since in our context, each measurement corresponds to a label, we obtain ranking problems of quite different complexity. Besides, even though the original measurements are real-valued, there are expression profiles containing ties which were broken randomly.

In order to complement the former real-world scenario with problems originating from several different domains, the following multiclass datasets from the UCI Repository of machine learning databases [6] and the Statlog collection [46] were included in the experimental evaluation: iris, wine, glass, vowel, vehicle (a summary of dataset properties is given in Table 2). These datasets were also used in a recent experimental study on multiclass support vector machines [32].

[8] We employed the implementation offered by the Weka machine learning package [61] in its default setting. To obtain a ranking of labels, classification scores were transformed into (pseudo-)probabilities using a logistic regression technique [50].

[9] This algorithm is based on the "alpha-bound trick". We set the corresponding parameter α to 500.

[10] Again, we used the implementation offered by the Weka package.

[11] All linear models also incorporate a bias term.

[12] This data is publicly available at http://www1.cs.columbia.edu/compbio/exp-phylo.

[13] We excluded three additional subproblems with more measurements due to the prohibitive computational demands of the constraint classification approach.

Table 3
Experimental results (mean and standard deviation) in terms of Kendall's tau

data	RPC	CC-P	CC-LR	CC-SVM	LL
iris	**.885 ± .068**	.836 ± .089	.836 ± .063	.812 ± .071	.818 ± .088
wine	.921 ± .053	.933 ± .043	.755 ± .111	.932 ± .057	**.942 ± .043**
glass	**.882 ± .042**	.846 ± .045	.834 ± .052	.820 ± .064	.817 ± .060
vowel	**.647 ± .019**	.623 ± .019	.583 ± .019	.594 ± .020	.601 ± .021
vehicle	.854 ± .025	**.855 ± .022**	.830 ± .025	.817 ± .025	.770 ± .037
spo	**.140 ± .023**	.138 ± .022	.122 ± .022	.121 ± .020	.132 ± .024
heat	.125 ± .024	**.126 ± .023**	.124 ± .024	.117 ± .023	.125 ± .025
dtt	.174 ± .034	**.180 ± .037**	.158 ± .033	.154 ± .045	.167 ± .034
cold	**.221 ± .028**	.220 ± .029	.196 ± .029	.193 ± .040	.209 ± .028
diau	**.332 ± .019**	.330 ± .019	.299 ± .022	.297 ± .019	.321 ± .020

Table 4
Experimental results (mean and standard deviation) in terms of Spearman's rank correlation

data	RPC	CC-P	CC-LR	CC-SVM	LL
iris	**.910 ± .058**	.863 ± .086	.874 ± .052	.856 ± .057	.843 ± .089
wine	.938 ± .045	.949 ± .033	.800 ± .102	.942 ± .052	**.956 ± .034**
glass	**.918 ± .036**	.889 ± .043	.879 ± .048	.860 ± .062	.859 ± .060
vowel	**.760 ± .020**	.746 ± .021	.712 ± .020	.724 ± .021	.732 ± .022
vehicle	.888 ± .020	**.891 ± .019**	.873 ± .022	.864 ± .023	.820 ± .036
spo	.176 ± .030	**.178 ± .030**	.156 ± .029	.156 ± .026	.167 ± .030
heat	.156 ± .030	**.156 ± .029**	.154 ± .029	.148 ± .027	.155 ± .031
dtt	.199 ± .040	**.205 ± .041**	.183 ± .038	.178 ± .054	.193 ± .038
cold	**.265 ± .033**	.265 ± .034	.234 ± .035	.235 ± .050	.251 ± .033
diau	**.422 ± .023**	.418 ± .023	.377 ± .026	.377 ± .022	.406 ± .025

For each of these four multiclass datasets, a corresponding ranking dataset was generated in the following manner: We trained a naive Bayes classifier[14] on the respective dataset. Then, for each example, *all* the labels present in the dataset were ordered with respect to decreasing predicted class probabilities (in the case of ties, labels with lower indices are ranked first). Thus, by substituting the single labels contained in the original multiclass datasets with the complete rankings, we obtain the label ranking datasets required for our experiments. The fundamental underlying learning problem may also be viewed as learning a qualitative replication of the probability estimates of a naive Bayes classifier.

6.2. Experimental results

6.2.1. Complete preference information

In the experiments, the actual true rankings on the test sets were compared to the corresponding predicted rankings. For each of the approaches, we report the average accuracy in terms of both Spearman's rank correlation and Kendall's tau. This is necessary because, as we showed in Section 5, RPC with weighted voting as a ranking procedure is especially tailored toward maximizing the Spearman rank correlation, while CC and LL are more focused on the Kendall tau measure: Minimization of the 0/1-loss on the expanded set of (binary) classification examples yields an implicit maximization of the empirical Kendall tau statistic of the label ranking function on the training set. It is true, however, that all distance (similarity) measures on rankings are of course more or less closely related.[15]

The results of a cross validation study (10-fold, 5 repeats), shown in Tables 3 and 4, are clearly in favor of RPC and CC in its online version. These two methods are on a par and outperform the other methods on all datasets except wine, for which LL yields the highest accuracy. These results are further corroborated by the standard classification accuracy on the multiclass data (probability to place the true class on the topmost rank), which is reported in Table 5.

In terms of training time, RPC is the clear winner, as can be seen in Table 6.[16] In compliance with our theoretical results, the original version of CC, here implemented as CC-SVM and CC-LR, was found to be quite problematic from this point of view, as it becomes extremely expensive for data sets with many attributes or many labels. For example, the trainings time for CC-SVM was almost 5 hours for vowel, and more than 7 days for the spo data; we therefore abstained from a detailed analysis and exposition of results for these variants. As expected, RPC is slightly less efficient than LL and CC-P in terms of

[14] We employed the implementation for naive Bayes classification on numerical datasets (NaiveBayesSimple) contained in the Weka machine learning package [61].

[15] For example, it has recently been shown in [14] that optimizing rank correlation yields a 5-approximation to the ranking which is optimal for the Kendall measure.

[16] Experiments were conducted on a PC Intel Core2 6600 2,4 Ghz with 2 GB RAM. We stopped the iteration in LL as soon as the sum of absolute changes of the weights was smaller than 10^{-7}; empirically, this was found to be the largest value that guaranteed stability of the model performance.

Table 5
Experimental results (mean and standard deviation) in terms of standard classification rate

data	RPC	CC-P	CC-LR	CC-SVM	LL
iris	**.952 ± .050**	.933 ± .069	.907 ± .075	.911 ± .076	.916 ± .076
wine	.945 ± .051	**.970 ± .042**	.927 ± .043	.948 ± .057	.962 ± .044
glass	**.767 ± .091**	.715 ± .089	.706 ± .092	.696 ± .099	.706 ± .093
vowel	**.507 ± .056**	.425 ± .062	.445 ± .063	.433 ± .064	.407 ± .067
vehicle	**.895 ± .028**	**.895 ± .034**	.868 ± .035	.865 ± .033	.851 ± .037

Table 6
Time (in ms) needed for training (left) and testing (mean and standard deviation)

data	RPC	CC-P	LL	RPC	CC-P	LL
iris	18 ± 11	48 ± 10	833 ± 587	0.6 ± 3.2	0.0 ± 0.0	0.0 ± 0.0
wine	59 ± 16	22 ± 14	575 ± 376	0.6 ± 3.1	0.3 ± 2.3	0.3 ± 2.3
glass	132 ± 15	605 ± 52	1529 ± 850	1.6 ± 4.8	0.0 ± 0.0	0.3 ± 2.3
vowel	927 ± 24	12 467 ± 595	36 063 ± 22 897	13.7 ± 5.1	0.3 ± 2.1	0.6 ± 3.1
vehicle	439 ± 24	1810 ± 177	2177 ± 1339	1.6 ± 4.8	0.0 ± 0.0	0.0 ± 0.0
spo	10 953 ± 95	343 506 ± 27190	61 826 ± 33 946	90.5 ± 5.8	0.9 ± 3.8	10.3 ± 8.1
heat	3069 ± 39	61 206 ± 3648	16 552 ± 9415	26.5 ± 7.3	0.6 ± 3.2	3.7 ± 6.7
dtt	1226 ± 31	19 592 ± 1133	2510 ± 1340	10.2 ± 7.4	0.3 ± 2.1	2.8 ± 6.0
cold	1209 ± 32	20 936 ± 1358	3045 ± 2001	10.6 ± 7.4	0.0 ± 0.0	3.4 ± 6.5
diau	4325 ± 38	83 967 ± 9849	27 441 ± 12 686	34.7 ± 6.6	1.2 ± 4.3	4.1 ± 7.0

testing time (see also Table 6), even though these times are extremely small throughout and clearly negligible in comparison with the training times.

6.2.2. Incomplete preference information

In Section 6.2.1, we provided an empirical study on learning label ranking functions assuming that the complete ranking is available for each example in the training set. However, in practical settings, we will often not have access to a total order of all possible labels for an object. Instead, in many cases, only a few pairs of preferences are known for each object.

To model incomplete preferences, we modified the training data as follows: A biased coin was flipped for every label in a ranking in order to decide whether to keep or delete that label; the probability for a deletion is p. Thus, a ranking such as $\lambda_1 \succ \lambda_2 \succ \lambda_3 \succ \lambda_4 \succ \lambda_5$ may be reduced to $\lambda_1 \succ \lambda_3 \succ \lambda_4$, and hence, pairwise preferences are generated only from the latter (note that, as a pairwise preference "survives" only with probability $(1-p)^2$, the average percentage of preferences in the training data decreases much faster with p than the average number of labels). Of course, the rankings produced in this way are of varying size.

Fig. 4 shows the experimental results for RPC, LL, and CC-P, the online variant of CC. More precisely, the figures show the accuracy in terms of Kendall's tau (which are qualitatively very similar to those for Spearman's rank correlation) as a function of the probability p. As expected, the accuracy decreases with an increasing amount of missing preference information, even though all three methods can deal with missing preference information remarkably well. Still, there seems to be a clear rank order: LL is the least sensitive method, and CC appears to be a bit less sensitive than RPC. Our explanation for this finding is that, due to training a quadratic instead of a linear number of models, RPC is in a sense more flexible than LL and CC. This flexibility is an advantage if enough training data is available but may turn out as a disadvantages if this is not the case. This may also explain the superior performance of LL on the wine data, which has relatively few instances. Finally, we mention that almost identical curves are obtained when sampling complete training examples with a suitable sampling rate. Roughly speaking, training on a few instances with complete preference information is comparable to training on more instances with partial preference information, provided the (expected) total number of pairwise preferences is the same.

7. Related work

As noted in Section 6, the work on constraint classification [28,29] appears to be a natural counterpart to our algorithm. In the same section, we have also discussed the log-linear models for label ranking proposed by Dekel et al. [17]. As both CC and LL are directly applicable to the label ranking problem studied in this paper, we compared RPC empirically with these approaches. The subsequent review will focus on other key works related to label ranking and pairwise decomposition techniques that have recently appeared in the literature; a somewhat more exhaustive literature survey can be found in [12].

We are not aware of any other work that, as our method, approaches the label ranking problem by learning pairwise preference predicates $\mathcal{R}_x(\lambda_i, \lambda_j)$, $1 \leqslant i < j \leqslant m$, and, thereby, reduces the problem to one of ranking on the basis of a preference relation. Instead, all existing methods, including CC and LL, essentially follow the idea of learning utility or scoring functions $f_1(\cdot) \dots f_m(\cdot)$ that can be used for inducing a label ranking: Given an input x, each label λ_i is evaluated in terms of a score $f_i(x)$, and the labels are then ordered according to these scores.

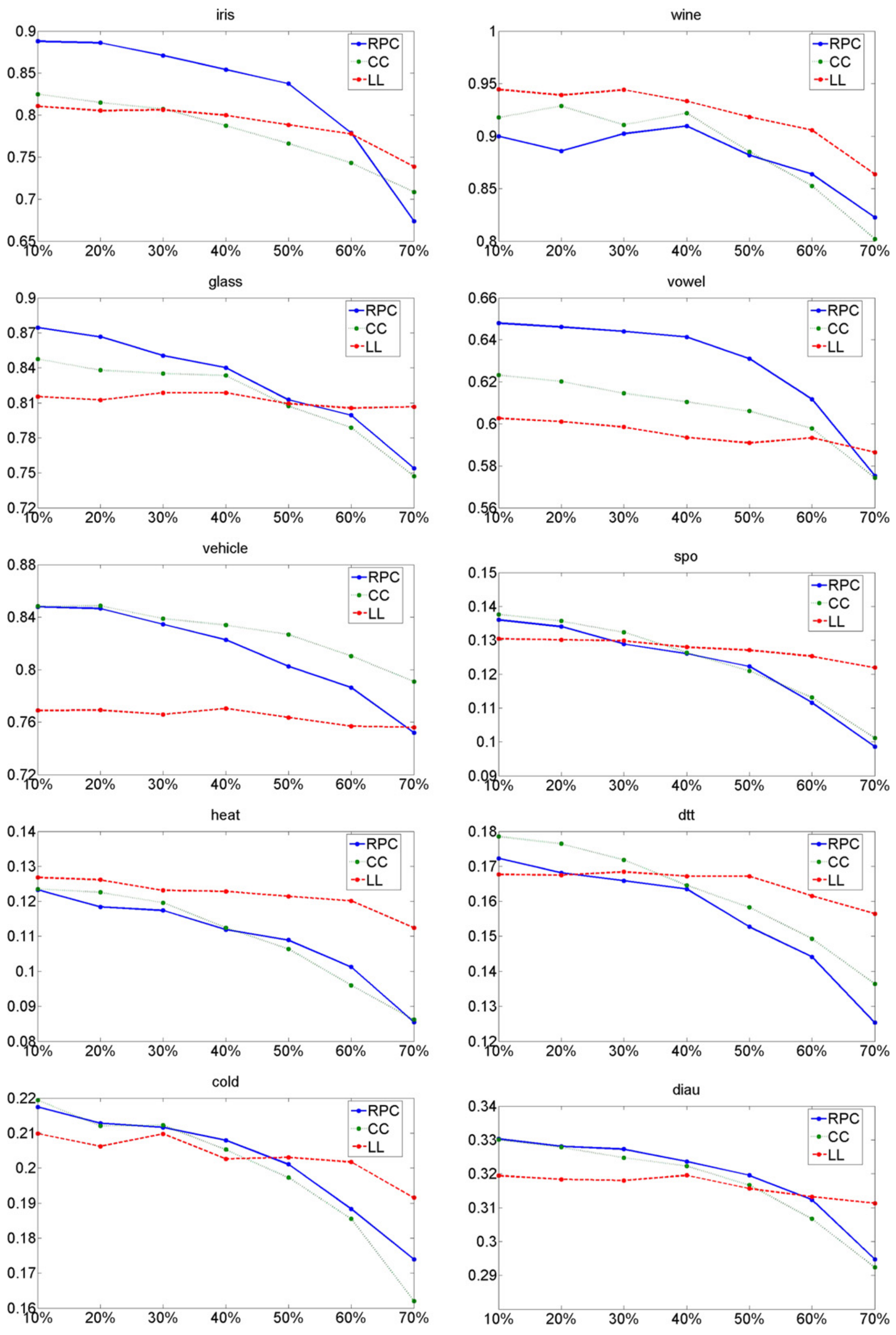

Fig. 4. Results for the datasets in Table 2 in the missing label scenario: Accuracy in terms of Kendall's tau as a function of the (expected) percentage of missing labels (note that different figures have different scales).

In passing, we note that, for the (important) special case in which we combine pairwise preferences in RPC by means of a simple voting strategy, it is true that we eventually compute a kind of score for each label as well, namely

$$f_i(x) = \sum_{1 \leqslant j \neq i \leqslant m} \mathcal{R}_x(\lambda_i, \lambda_j), \tag{7.1}$$

that may, at least at first sight, appear comparable to the utility functions

$$f_i(x) = \sum_j \alpha_j h_j(x, \lambda_i) \tag{7.2}$$

used in LL. However, despite a formal resemblance, one should note that (7.1) is not directly comparable to (7.2). In particular, our "base functions" are preference predicates ($\mathcal{L} \times \mathcal{L} \to [0, 1]$ mappings) instead of scoring functions ($\mathcal{X} \times \mathcal{L} \to \mathbb{R}$ mappings). Moreover, as opposed to (7.2), the number of these functions is predetermined by the number of labels (m), and each of them has the same relevance (i.e., weighing coefficients α_i are not needed).

Shalev-Shwartz and Singer [56] learn utility functions $f_i(\cdot)$ on the basis of a different type of training information, namely real values $g(\lambda_i)$ that reflect the relevance of the labels λ_i for an input x. Binary preferences between labels λ_i and λ_j are then weighted by the difference $g(\lambda_i) - g(\lambda_j)$, and this value is considered as a degree of importance of ordering λ_i ahead of λ_j. This framework hence deviates from a purely qualitative setting in which preference information is modeled in the form of order relations.

Another interesting generalization of the utility-based approach to label ranking is the framework of Aiolli [1], that allows one to specify both qualitative and quantitative preference constraints on utility functions. In addition to the pairwise preference constraints that we also use (and which he interprets as constraints on a utility function), Aiolli [1] also allows constraints of the type $\lambda_i \succeq_x \tau$, which means that the value of the utility function $f_i(x) > t_i$, where t_i is a numerical threshold.

There has also been some previous work on the theoretical foundations of label ranking. We already mentioned above that Dekel et al. [17] introduced a generalized ranking error, which assumes a procedure for decomposing a preference graph into subgraphs, and defines the generalized error as the fraction of subgraphs that are not exactly in agreement with the learned utility function. Ha and Haddawy [26] discuss a variety of different ranking loss functions and introduce a different extension of Kendall's tau. With respect to predictive performance, Usunier et al. [59] analyze the generalization properties of binary classifiers trained on interdependent data for certain types of structured learning problems such as bipartite ranking.

As mentioned in Section 2, label ranking via pairwise preference models may be viewed as a generalization of various other learning tasks. There has been a considerable amount of recent work on many of such tasks. In particular, pairwise classification has been studied in-depth in the area of support vector machines [32, and references therein]. We refer to [22, Section 8] for a brief survey of work on pairwise classification, and its relation to other learning class binarization techniques.

Another special scenario is the application of label ranking algorithms to multi-label problems. For example, Crammer and Singer [16] consider a variety of on-line learning algorithms for the problem of ranking possible labels in a multi-label text categorization task. They investigate a set of algorithms that maintain a prototype for each possible label, and order the labels of an example according to the response signal returned by each of the prototypes. [11] demonstrates a general technique that not only allows one to rank all possible labels in multi-label problem, but also to select an appropriate threshold between relevant and irrelevant labels.

It is well-known that pairwise classification is a special case of Error Correcting Output Codes (ECOC) [18] or, more precisely, their generalization that has been introduced in [2]. Even though ECOC allows for a more flexible decomposition of the original problem into simpler ones, the pairwise approach has the advantage that it provides a fixed, domain-independent and non-stochastic decomposition with a good overall performance. In several experimental studies, including [2], it performed en par or better with competing decoding matrices. While finding a good encoding matrix still is an open problem [49], it can be said that pairwise classification is among the most efficient decoding schemes. Even though we have to train a quadratic number of classifiers, both training (and to some extent also testing) can be performed in linear time as discussed in Section 4. ECOC matrices that produce the necessary redundancy by defining more binary prediction problems than labels are more expensive to train.

What is more important here, however, is that the pairwise case seems to have special advantages in connection with *ranking and preference learning problems*. In particular, it has a clearly defined semantics in terms of pairwise comparison between alternatives and, as we discussed in Section 3, produces as output a binary *preference relation*, which is an established concept in preference modeling and decision theory. As opposed to this, the semantics of a model that compares more than two classes, namely a subset of positive with a subset of negative ones, as it is possible in ECOC, is quite unclear. For example, while a prediction $\lambda_3 \succ \lambda_2$ obviously indicates that λ_3 is ranked before λ_2, several interpretations are conceivable for a prediction such as, say, $\{\lambda_3, \lambda_5\} \succ \{\lambda_1, \lambda_2\}$. Without going into further detail, we mention that all these interpretations seem to produce serious complications, either with regard to the training of models or the decoding step, or both. In any case, generalizing the pairwise approach in the label ranking setting appears to be much more difficult than in the classification setting, where an information about class membership can easily be generalized from single labels (the instance belongs to λ_3) to a set of labels (the instance belongs to λ_3 or λ_5). The main reason is that, in label ranking, a single piece of information does not concern a class membership but preference (order) information that naturally relates to pairs of labels.

8. Conclusions

In this paper, we have introduced a learning algorithm for the *label ranking* problem and investigated its properties both theoretically and empirically. The merits of our method, called ranking by pairwise comparison (RPC), can be summarized as follows:

- Firstly, we find that RPC is a simple yet intuitively appealing and elegant approach, especially as it is a natural generalization of pairwise classification. Besides, RPC is completely in line with preference modeling based on binary preference relations, an established approach in decision theory.
- Secondly, the modular conception of RPC allows for combining different (pairwise) learning and ranking methods in a convenient way. For example, different loss functions can be minimized by simply changing the ranking procedure but without the need to retrain the binary models (see Section 5).
- Thirdly, RPC is superior to alternative approaches with regard to efficiency and computational complexity, as we have shown both theoretically and experimentally (cf. Sections 4 and 6), while being at least competitive in terms of prediction quality.
- Fourthly, while existing label ranking methods are inherently restricted to linear models, RPC is quite general regarding the choice of a base learner, as in principle every binary classifier can be used.

Finally, we note that RPC also appears attractive with regard to an extension of the label ranking problem to the learning of more general preference relations on the label set $\mathcal{L}$. In fact, in many practical applications it might be reasonable to relax the assumption of strictness, i.e., to allow for indifference between labels, or even to represent preferences in terms of *partial* instead of total orders. The learning of pairwise preference predicates is then definitely more suitable than utility-based methods, since a utility function necessarily induces a total order and, therefore, cannot represent partial orders. Extensions of this kind constitute important aspects of ongoing work.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments that helped to considerably improve this paper. This research was supported by the German Research Foundation (DFG).

Appendix A. Transitivity properties of pairwise preferences

Our pairwise learning scheme introduced in Section 3 produces a preference relation $\mathcal{R}_x$ in a first step, which is then used for inducing a ranking τ_x. As *transitivity* of pairwise preferences is one of the most important properties in preference modeling, an interesting question is whether any sort of transitivity can be guaranteed for $\mathcal{R}_x$. Indeed, even though the pairwise preferences induced by a *single* ranking are obviously transitive, it is less clear whether this property is preserved when "merging" different rankings in a probabilistic way.

In fact, recall that every instance $x \in \mathcal{X}$ is associated with a probability distribution over $\mathcal{S}_m$ (cf. Section 5.1). Such a distribution induces a unique probability distribution for pairwise preferences via

$$p_{ij} = \mathbb{P}(\lambda_i \succ \lambda_j) = \sum_{\tau \in \mathcal{S}_m:\ \tau(i) < \tau(j)} \mathbb{P}(\tau). \tag{A.1}$$

An interesting finding is that the pairwise preferences (A.1) do indeed satisfy a form of transitivity, albeit a relatively weak one:

$$\forall i, j, k \in \{1 \dots m\}:\ p_{ik} \geqslant p_{ij} + p_{jk} - 1. \tag{A.2}$$

More formally, we can prove the following theorem.

Theorem 4. *Consider any probability distribution on the set of rankings $\mathcal{S}_m$. The pairwise preferences induced by this distribution via* (A.1) *satisfy* (A.2).

Proof. Consider any three labels $\lambda_i, \lambda_j, \lambda_k$. Obviously, there is no need to distinguish the rankings which put these labels in the same order. Thus, we can partition $\mathcal{S}_m$ into six equivalence classes $S_{ijk}, S_{ikj} \dots S_{kij}$, where $S_{ijk} = \{\tau \in \mathcal{S}_m \mid \tau(i) < \tau(j) < \tau(k)\}$ and the other classes are defined analogously. Let

$$q_{ijk} \overset{\mathrm{df}}{=} \mathbb{P}(S_{ijk}) = \sum_{\tau \in \mathcal{S}_m:\ \tau(i) < \tau(j) < \tau(k)} \mathbb{P}(\tau)$$

and $q = (q_{ijk}, q_{ikj}, q_{jik}, q_{jki}, q_{kij}, q_{kji})^\top \in [0,1]^6$.

Now, consider probabilities $p = (p_{ij}, p_{jk}, p_{ik})^\top$ for the pairwise probabilities (A.1). Finding a distribution on rankings which induces these probabilities obviously comes down to solving a system of linear equations of the form $A \times q = p$, where A is a matrix of dimension 3×6 with 0/1 entries, and

$$q_{ijk} + q_{ikj} + q_{jik} + q_{jki} + q_{kij} + q_{kji} = 1.$$

The set of solutions to this problem can be expressed as

$$\begin{pmatrix} q_{ijk} \\ q_{ikj} \\ q_{jik} \\ q_{jki} \\ q_{kij} \\ q_{kji} \end{pmatrix} = \begin{pmatrix} p_{ij} + p_{jk} - 1 + v \\ 1 - p_{jk} - u - v \\ p_{ik} - p_{ij} + u \\ 1 - p_{ik} - u - v \\ u \\ v \end{pmatrix}$$

where $u, v \in [0, 1]$. Additionally, the components of q must be non-negative. If this is satisfied for $u = v = 0$, then $p_{ik} \geqslant p_{ij}$ (fourth entry) and (A.2) holds. In the case where non-negativity is violated, either $p_{ij} + p_{jk} < 1$ or $p_{ik} < p_{ij}$. In the second case, u must be increased to (at least) $p_{ij} - p_{ik}$, and one obtains the solution vector

$$\left(p_{ij} + p_{jk} - 1,\ 1 + p_{ik} - (p_{ij} + p_{jk}),\ 0,\ 1 - p_{ij},\ p_{ij} - p_{ik},\ 0\right)^\top$$

which is non-negative if and only if $p_{ik} \geqslant p_{ij} + p_{jk} - 1$. In the first case, v must be increased to (at least) $1 - (p_{ij} + p_{jk})$, and one obtains the solution vector

$$\left(0,\ p_{ij},\ p_{ik} - p_{ij},\ p_{ij} + p_{jk} - p_{ik},\ 0,\ 1 - (p_{ij} + p_{jk})\right)^\top$$

which is non-negative if and only if $p_{ik} \leqslant p_{ij} + p_{jk}$. This latter inequality is equivalent to $p_{kj} \geqslant p_{kj} + p_{ji} - 1$, where $p_{kj} = 1 - p_{jk}$, so the transitivity property (A.2) now holds for the reciprocal probabilities. In a similar way one verifies that (A.2) must hold in the case where both $p_{ij} + p_{jk} < 1$ and $p_{ik} < p_{ij}$. In summary, a probability distribution on $\mathcal{S}_m$ which induces the probabilities p_{ij}, p_{jk}, p_{ik} exists if and only if these probabilities satisfy (A.2). $\square$

It is interesting to note that (A.2) is a special type of $\top$-transitivity. A so-called t-norm is a generalized logical conjunction, namely a binary operator $\top : [0, 1]^2 \to [0, 1]$ which is associative, commutative, monotone, and satisfies $\top(0, x) = 0$, $\top(1, x) = x$ for all x. Operators of that kind have been introduced in the context of probabilistic metric spaces [55] and have been studied intensively in fuzzy set theory in recent years [39]. A binary relation $\mathcal{R} \subset A \times A$ is called $\top$-transitive if it satisfies $\mathcal{R}(a, c) \geqslant \top(\mathcal{R}(a, b), \mathcal{R}(b, c))$ for all $a, b, c \in A$. Therefore, what the condition (A.2) expresses is just $\top$-transitivity with respect to the Lukasiewicz t-norm which is defined by $\top(x, y) = \max(x + y - 1, 0)$. An interesting idea to guarantee this condition to hold is hence to replace the original ensemble of pairwise predictions by its $\top$-transitive closure [47], where $\top$ is the aforementioned Lukasiewicz t-norm.

References

[1] Fabio Aiolli, A preference model for structured supervised learning tasks, in: Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM-05), IEEE Computer Society, 2005, pp. 557–560.

[2] Erin L. Allwein, Robert E. Schapire, Yoram Singer, Reducing multiclass to binary: A unifying approach for margin classifiers, Journal of Machine Learning Research 1 (2000) 113–141.

[3] Noga Alon, Ranking tournaments, SIAM Journal on Discrete Mathematics 20 (1), pp. 137–142.

[4] Rajarajeswari Balasubramaniyan, Eyke Hüllermeier, Nils Weskamp, Jörg Kämper, Clustering of gene expression data using a local shape-based similarity measure, Bioinformatics 21 (7) (2005) 1069–1077.

[5] John J. Bartholdi III, Craig A. Tovey, Michael A. Trick, Voting schemes for which it can be difficult to tell who won the election, Social Choice and Welfare 6 (2) (1989) 157–165.

[6] Catherine L. Blake, Christopher J. Merz, UCI repository of machine learning databases, 1998; Data available at http://www.ics.uci.edu/~mlearn/MLRepository.html.

[7] Craig Boutilier, Ronen Brafman, Carmel Domshlak, Holger Hoos, David Poole, CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements, Journal of Artificial Intelligence Research 21 (2004) 135–191.

[8] Ralph A. Bradley, Milton E. Terry, The rank analysis of incomplete block designs—I. The method of paired comparisons, Biometrika 39 (1952) 324–345.

[9] Steven J. Brams, Peter C. Fishburn, Voting procedures, in: K.J. Arrow, A.K. Sen, K. Suzumura (Eds.), Handbook of Social Choice and Welfare (vol. 1), Elsevier, 2002, Chapter 4.

[10] Pavel B. Brazdil, Carlos Soares, J.P. da Costa, Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results, Machine Learning 50 (3) (March 2003) 251–277.

[11] Klaus Brinker, Johannes Fürnkranz, Eyke Hüllermeier, A unified model for multilabel classification and ranking, in: Proceedings of the 17th European Conference on Artificial Intelligence (ECAI-06), 2006, pp. 489–493.

[12] Klaus Brinker, Johannes Fürnkranz, Eyke Hüllermeier, Label ranking by learning pairwise preferences, Technical Report TUD-KE-2007-01, Knowledge Engineering Group, TU Darmstadt, 2007.

[13] William W. Cohen, Robert E. Schapire, Yoram Singer, Learning to order things, Journal of Artificial Intelligence Research 10 (1999) 243–270.

[14] Don Coppersmith, Lisa Fleischer, and Atri Rudra, Ordering by weighted number of wins gives a good ranking for weighted tournaments, in: Proceedings of the ACM–SIAM Symposium on Discrete Algorithms (SODA), 2006, pp. 776–782.

[15] Koby Crammer, Yoram Singer, Ultraconservative online algorithms for multiclass problems, Journal of Machine Learning Research 3 (2003) 951–991.

[16] Koby Crammer, Yoram Singer, A family of additive online algorithms for category ranking, Journal of Machine Learning Research 3 (2003) 1025–1058.
[17] Ofer Dekel, Christopher D. Manning, Yoram Singer, Log-linear models for label ranking, in: S. Thrun, L.K. Saul, B. Schölkopfand (Eds.), Advances in Neural Information Processing Systems 16 (NIPS-2003), MIT Press, 2004.
[18] Thomas G. Dietterich, Ghulum Bakiri, Solving multiclass learning problems via error-correcting output codes, Journal of Artificial Intelligence Research 2 (1995) 263–286.
[19] Jon Doyle, Prospects for preferences, Computational Intelligence 20 (2) (2004) 111–136.
[20] János Fodor, Marc Roubens, Fuzzy Preference Modelling and Multicriteria Decision Support, Kluwer Academic Publishers, 1994.
[21] Jerome H. Friedman, Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University, Stanford, CA, 1996.
[22] Johannes Fürnkranz, Round robin classification, Journal of Machine Learning Research 2 (2002) 721–747.
[23] Johannes Fürnkranz, Round robin ensembles, Intelligent Data Analysis 7 (5) (2003) 385–404.
[24] Johannes Fürnkranz, Eyke Hüllermeier, Pairwise preference learning and ranking, in: N. Lavrač, D. Gamberger, H. Blockeel, L. Todorovski (Eds.), Proceedings of the 14th European Conference on Machine Learning (ECML-03), Cavtat, Croatia, in: Lecture Notes in Artificial Intelligence, vol. 2837, Springer-Verlag, 2003.
[25] Johannes Fürnkranz, Eyke Hüllermeier, Preference learning, Künstliche Intelligenz 19 (1) (2005) 60–61.
[26] Vu Ha, Peter Haddawy, Similarity of personal preferences: Theoretical foundations and empirical analysis, Artificial Intelligence 146 (2003) 149–173.
[27] Peter Haddawy, Vu Ha, Angelo Restificar, Benjamin Geisler, John Miyamoto, Preference elicitation via theory refinement, Journal of Machine Learning Research 4 (2003) 317–337.
[28] Sariel Har-Peled, Dan Roth, Dav Zimak, Constraint classification: A new approach to multiclass classification, in: N. Cesa-Bianchi, M. Numao, R. Reischuk (Eds.), Proceedings of the 13th International Conference on Algorithmic Learning Theory (ALT-02), Lübeck, Germany, Springer, 2002, pp. 365–379.
[29] Sariel Har-Peled, Dan Roth, Dav Zimak, Constraint classification for multiclass classification and ranking, in: Suzanna Becker, Sebastian Thrun, Klaus Obermayer (Eds.), Advances in Neural Information Processing Systems 15 (NIPS-02), 2003, pp. 785–792.
[30] Trevor Hastie, Robert Tibshirani, Classification by pairwise coupling, in: M.I. Jordan, M.J. Kearns, S.A. Solla (Eds.), Advances in Neural Information Processing Systems 10 (NIPS-97), MIT Press, 1998, pp. 507–513.
[31] Ralf Herbrich, Thore Graepel, Peter Bollmann-Sdorra, Klaus Obermayer, Supervised learning of preference relations, in: Proceedings des Fachgruppentreffens Maschinelles Lernen (FGML-98), 1998, pp. 43–47.
[32] Chih-Wei Hsu, Chih-Jen Lin, A comparison of methods for multi-class support vector machines, IEEE Transactions on Neural Networks 13 (2) (March 2002) 415–425.
[33] Eyke Hüllermeier, Johannes Fürnkranz, Ranking by pairwise comparison: A note on risk minimization, in: Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE-04), Budapest, Hungary, 2004.
[34] Eyke Hüllermeier, Johannes Fürnkranz, Comparison of ranking procedures in pairwise preference learning, in: Proceedings of the 10th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU-04), Perugia, Italy, 2004.
[35] Thorsten Joachims, Optimizing search engines using clickthrough data, in: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-02), ACM Press, 2002, pp. 133–142.
[36] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, Geri Gay, Accurately interpreting clickthrough data as implicit feedback, in: Proceedings of the 28th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR-05), 2005.
[37] Roni Khardon, Gabriel Wachman, Noise tolerant variants of the perceptron algorithm, The Journal of Machine Learning Research 8 (2007) 227–248.
[38] Maurice G. Kendall, Rank Correlation Methods, Charles Griffin, London, 1955.
[39] Erich-Peter Klement, Radko Mesiar, Endre Pap, Triangular Norms, Kluwer Academic Publishers, 2002.
[40] Stefan Knerr, Léon Personnaz, Gérard Dreyfus, Single-layer learning revisited: A stepwise procedure for building and training a neural network, in: F. Fogelman Soulié, J. Hérault (Eds.), Neurocomputing: Algorithms, Architectures and Applications, in: NATO ASI Series, vol. F68, Springer-Verlag, 1990, pp. 41–50.
[41] Stefan Knerr, Léon Personnaz, Gérard Dreyfus, Handwritten digit recognition by neural networks with single-layer training, IEEE Transactions on Neural Networks 3 (6) (1992) 962–968.
[42] Ulrich H.-G. Kreßel, Pairwise classification and support vector machines, in: B. Schölkopf, C.J.C. Burges, A.J. Smola (Eds.), Advances in Kernel Methods: Support Vector Learning, MIT Press, Cambridge, MA, 1999, pp. 255–268, Chapter 15.
[43] Erich L. Lehmann, H.J.M. D'Abrera, Nonparametrics: Statistical Methods Based on Ranks, rev. ed., Prentice-Hall, Englewood Cliffs, NJ, 1998.
[44] Bao-Liang Lu, Masami Ito, Task decomposition and module combination based on class relations: A modular neural network for pattern classification, IEEE Transactions on Neural Networks 10 (5) (September 1999) 1244–1256.
[45] John I. Marden, Analyzing and Modeling Rank data, Chapman & Hall, London, 1995.
[46] Donald Michie, David J. Spiegelhalter, C.C. Taylor, Machine Learning, Neural and Statistical Classification, Ellis Horwood, 1994, Data available at ftp://ftp.ncc.up.pt/pub/statlog/.
[47] Helga Naessens, Hans De Meyer, Bernard De Baets, Algorithms for the computation of T-transitive closures, IEEE Trans. Fuzzy Syst. 10 (2002) 541–551.
[48] Sang-Hyeun Park, Johannes Fürnkranz, Efficient Pairwise Classification, in: Proceedings of the 17th European Conference on Machine Learning (ECML-07), Warsaw, Poland, Springer-Verlag, September 2007, pp. 658–665.
[49] Edgar Pimenta, João Gama, André Carvalho, Pursuing the best ECOC dimension for multiclass problems, in: Proceedings of the 20th International Florida Artificial Intelligence Research Society Conference (FLAIRS-07), 2007, pp. 622–627.
[50] John Platt, Probabilistic outputs for support vector machines and comparison to regularized likelihood methods, in: A.J. Smola, P. Bartlett, B. Schoelkopf, D. Schuurmans (Eds.), Advances in Large Margin Classifiers, Cambridge, MA, MIT Press, 1999, pp. 61–74.
[51] David Price, Stefan Knerr, Léon Personnaz, Gérard Dreyfus, Pairwise neural network classifiers with probabilistic outputs, in: G. Tesauro, D. Touretzky, T. Leen (Eds.), Advances in Neural Information Processing Systems 7 (NIPS-94), MIT Press, 1995, pp. 1109–1116.
[52] Filip Radlinski, Thorsten Joachims, Learning to rank from implicit feedback, in: Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD-05), 2005.
[53] Ryan Rifkin, Aldebaro Klautau, In defense of one-vs-all classification, Journal of Machine Learning Research 5 (2004) 101–141.
[54] Michael S. Schmidt, Herbert Gish, Speaker identification via support vector classifiers, in: Proceedings of the 21st IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-96), Atlanta, GA, 1996, pp. 105–108.
[55] B. Schweizer, A. Sklar, Probabilistic Metric Spaces, North-Holland, New York, 1983.
[56] Shai Shalev-Shwartz, Yoram Singer, Efficient learning of label ranking by soft projections onto polyhedra, Journal of Machine Learning Research 7 (2006) 1567–1599.
[57] Charles Spearman, The proof and measurement of association between two things, American Journal of Psychology 15 (1904) 72–101.
[58] Gerald Tesauro, Connectionist learning of expert preferences by comparison training, in: D. Touretzky (Ed.), Advances in Neural Information Processing Systems 1 (NIPS-88), Morgan Kaufmann, 1989, pp. 99–106.
[59] Nicolas Usunier, Massih-Reza Amini, Patrick Gallinari, Generalization error bounds for classifiers trained with interdependent data, in: Y. Weiss, B. Schölkopf, J. Platt (Eds.), Advances in Neural Information Processing Systems 18 (NIPS 2005), MIT Press, 2006, pp. 1369–1376.

[60] Jun Wang, Artificial neural networks versus natural neural networks: A connectionist paradigm for preference assessment, Decision Support Systems 11 (1994) 415–429.
[61] Ian H. Witten, Eibe Frank, Data Mining: Practical Machine Learning Tools with Java Implementations, Morgan Kaufmann, San Francisco, 2000.
[62] Ting-Fan Wu, Chih-Jen Lin, Ruby C. Weng, Probability estimates for multi-class classification by pairwise coupling, Journal of Machine Learning Research 5 (Aug) (2004) 975–1005.