Technische Universität Darmstadt
Knowledge Engineering Group
Hochschulstrasse 10, D-64289 Darmstadt, Germany

http://www.ke.informatik.tu-darmstadt.de

Technical Report TUD–KE–2007–01

Klaus Brinker, Johannes Fürnkranz, Eyke Hüllermeier

Label Ranking by Learning Pairwise Preferences

Label Ranking by Learning Pairwise Preferences

Klaus Brinker BRINKER@ITI.CS.UNI-MAGDEBURG.DE
Data and Knowledge Engineering Group
Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg, Germany

Johannes Fürnkranz FUERNKRANZ@INFORMATIK.TU-DARMSTADT.DE
Knowledge Engineering Group
Department of Computer Science
TU Darmstadt, Germany

Eyke Hüllermeier HUELLERM@ITI.CS.UNI-MAGDEBURG.DE
Data and Knowledge Engineering Group
Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg, Germany

Abstract

Preference learning is a challenging problem that involves the prediction of complex structures, such as weak or partial order relations. In the recent literature, the problem appears in many different guises, which we will first put into a coherent framework. This work then 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 ranking function, 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 learned relation by means of a ranking procedure, whereby different ranking methods can be used for minimizing different loss functions. In particular, we show that (weighted) voting as a rank aggregation technique minimizes the Spearman rank correlation. Finally, we compare RPC to constraint classification, an alternative approach to label ranking, and show empirically and theoretically that RPC is computationally more efficient.

1. Introduction

Recently, the topic of *preferences* has attracted considerable attention in Artificial Intelligence (AI) research, notably in fields such as agents, non-monotonic reasoning, constraint satisfaction, planning, and qualitative decision theory (Doyle, 2004).[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. Therefore, the formal modeling of preferences can be considered an essential aspect of autonomous agent design.

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).

Drawing on past research on knowledge representation and reasoning, AI offers qualitative and symbolic methods for treating preferences that can reasonably complement traditional approaches that have been developed for quite a while in fields such as economic decision theory. Needless to say, however, the acquisition of preferences is not always an easy task. Therefore, not only are modeling languages and representation formalisms needed, but also methods for the automatic learning, discovery and adaptation of preferences. For example, computerized methods for discovering the preferences of individuals are useful in e-commerce and various other fields where an increasing trend toward personalization of products and services can be recognized.

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. In Section 2, we discriminate between *object ranking problems*, where the task is to order a set of objects according to a preference function, and *label ranking problems*, where the task is to assign a permutation of a fixed set of labels to a given object.

Subsequently, we observe that there are two principal approaches to address preference learning tasks. One possibility is to learn a *utility function* which induces the sought ranking by *scoring individual objects*. The assumption behind this approach is that the observed preferences are based on a hidden (latent) numerical preference model, that produces a numerical score for each object or label. The alternative is to *compare pairs of objects*, that is, to learn a *binary preference relation*. The assumption here is that for deriving a total order, it suffices to be able to predict which one of each pair of objects or labels is preferable. While this approach sounds logical, it is complicated by the fact that preference predicates learned from data are typically not transitive and consequently do not induce a natural total order.

Out of the resulting four groups of problem categories (each combination of the two learning tasks and two modeling approaches), three have already been addressed in the literature, while the fourth is the subject of this paper. More specifically, the learning scenario that we will consider in this paper consists of 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 consistent), each one expressing that one label is preferred over another. The goal is to use pairwise preferences from training examples for predicting a total order, a *ranking*, of all possible labels for a new training example. The *ranking by pairwise comparison* (RPC) algorithm, that we introduce and investigate in this paper, has a modular structure 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*.

The remainder of the paper is organized as follows. Section 2 gives a more thorough introduction to preference learning and a systematic overview of learning problems and existing approaches in this field. Our learning algorithm (RPC) is described in detail in Section 3. Formal properties of RPC, including complexity and optimality issues, are discussed in Section 4. Finally, Section 5 is devoted to an experimental evaluation of RPC and a comparison with an alternative approach applicable to the same learning problem. Parts of this paper are based on (Fürnkranz and Hüllermeier, 2003; 2005; Hüllermeier and Fürnkranz, 2004a).

1.1 Notation

A binary relation $\mathcal{R}$ on a set $\mathcal{A}$ of objects (alternatives) is a subset of $\mathcal{A} \times \mathcal{A}$. For $a, b \in \mathcal{A}$, $\mathcal{R}(a, b)$ means that a is related to b, i.e., the tuple (a, b) belongs to the relation $\mathcal{R}$. In agreement with our preference semantics, we shall also write $a \succeq b$ which is interpreted as "alternative a is at least as good as alternative b". A weak preference relation $\succeq$ induces a strict preference relation $\succ$ and an indifference relation $\sim$ as follows: $a \succ b$ iff $(a \succeq b)$ and $(b \not\succeq a)$; moreover, $a \sim b$ iff $(a \succeq b)$ and $(b \succeq a)$.

A relation $\succeq$ is *reflexive* if $a \succeq a$ for all $a \in \mathcal{A}$, *irreflexive* if $a \not\succeq a$ for all a, *total* if $a \succeq b$ or $b \succeq a$ for all $a \neq b$, *antisymmetric* if $a \succeq b$ and $b \succeq a$ implies $a = b$, *transitive* if $a \succeq b$ and $b \succeq c$ implies $a \succeq c$ for all $a, b, c \in \mathcal{A}$.

A relation $\mathcal{R}$ is a *total strict order (ranking)* if it is total, irreflexive and transitive (and hence antisymmetric). Typically, we will use the symbol $\succ$ to denote a ranking. If $\mathcal{A}$ is a finite set $\{a_1 \ldots a_m\}$, a ranking can be identified with a permutation τ of $\{1 \ldots m\}$. In fact, given a ranking $\succ$ on $\mathcal{A}$, there is a unique permutation τ such that $a_i \succ a_j$ if and only if $\tau(i) < \tau(j)$ ($\tau(i)$ is the position of a_i in the ranking). We shall denote the class of all permutations of $\{1 \ldots m\}$ by $\mathcal{S}_m$. Moreover, by abuse of notation, we shall sometimes employ the terms "ranking" and "permutation" synonymously.

We use the symbol $\mathbb{R}$ for the set of real numbers, and $\mathbb{P}(\cdot)$ for probabilities.

2. Learning from Preferences

In this section, we will motivate preference learning 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, or by *comparing* (pairs of) competing alternatives. All four combinations of these possibilities are possible (cf. Table 1). 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.

Before discussing these four settings in more detail, let us note that the term "preference" should not be taken literally and instead always be interpreted 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.

	modeling utility functions	modeling pairwise preferences
object ranking	comparison training (Tesauro, 1989)	learning to order things (Cohen et al., 1999)
label ranking	constraint classification (Har-Peled et al., 2002)	**this work** (Fürnkranz and Hüllermeier, 2003)

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

---

Given:

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

Find:

- a ranking function $r(\cdot)$ that returns a permutation (ranking) of each set of objects $\mathcal{O} \subseteq \mathcal{X}$

Figure 1: Learning from object preferences

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 Figure 1, is also known as "learning to order things" (Cohen et al., 1999).

As an example consider the problem of learning to rank query results of a search engine (Joachims, 2002; Radlinski and Joachims, 2005). 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 (Joachims et al., 2005).

Given:

- a set of training instances $\{x_k \mid k = 1 \dots n\} \subseteq \mathcal{X}$
  (each instance typically represented by a feature vector)
- a set of labels $\mathcal{L} = \{\lambda_i \mid i = 1 \dots m\}$
- 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 $\tau_x \in \mathcal{S}_m$)

Figure 2: Learning from label preferences

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, as we are especially interested in the case where $\succ_x$ is a total strict order, the problem is to predict a permutation of $\mathcal{L}$. The training information consists of a set of instances for which (partial) knowledge about the associated preference relation is available (cf. Figure 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.

It has been observed by several authors (Har-Peled et al., 2002; Fürnkranz and Hüllermeier, 2003; Dekel et al., 2004) that the above setting may be viewed as a generalization of several standard learning settings. In particular, the following problems are special cases of learning label preferences:

- *Classification:* A single class label λ_i is assigned to each example x_k. This implicitly defines the set of preferences $\{\lambda_i \succ_{x_k} \lambda_j \mid 1 \leq j \neq i \leq m\}$.
- *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 $\{\lambda_i \succ_{x_k} \lambda_j \mid \lambda_i \in L_k, \lambda_j \in \mathcal{L} \setminus L_k\}$.
- *Ranking:* Each training example x_k is associated with a ranking (total strict order) of the labels.

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 postprocessing step, for example a projection to the top-rank in classification learning where only this label is relevant.

As in the case of object ranking, the learning scenario introduced in this section has a large number of practical applications. To illustrate, consider the problem of tailoring the order of questions in a questionnaire to the characteristics of a particular respondent, with the motivation to to maximize his motivation to complete the questionnaire. Another example is learning to predict the best order in which to supply a certain set of stores (i.e., the route of a truck), depending on external conditions like, e.g., traffic, weather, purchase order quantities (which define the features of the instance). A challenging learning scenario, which is also included in the empirical evaluation study, originates from the bioinformatics field where structured data can frequently be found. 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 (Balasubramaniyan et al., 2005). Yet 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 (Brazdil et al., 2003). Finally, every preference statement in the well-known CP-nets approach Boutilier et al. (2004), a qualitative graphical representation that reflects conditional dependence and independence of preferences under a *ceteris paribus* interpretation, formally corresponds to a label ranking.

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, $i = 1 \dots m$. 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., $\lambda_i \succeq_x \lambda_j \Leftrightarrow f_i(x) \geq f_j(x)$.

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 (1989) 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. Similarly, Haddawy et al. (2003) use training data in the form of pairwise comparisons of objects to train a neural network. The network learns a function that takes two objects as input and outputs either

0 or 1, depending on whether or not the first object is preferred to the second one. The structure of the network is not arbitrary, but is chosen in a way that allows one to combine domain knowledge about partial preferences between the feature-based state descriptions of the objects. This work is based on an earlier approach by Wang (1994).

Similar ideas have also been investigated for training other types of classifiers, in particular support vector machines. We already mentioned Joachims (2002) who analyzed "click-through data" in order to rank documents retrieved by a search engine according to their relevance. Earlier, Herbrich et al. (1998) have proposed an algorithm for training SVMs from pairwise preference relations between objects.

For the case of label ranking, a corresponding method for learning the functions $f_i(\cdot)$, $i = 1 \dots m$, from training data has been proposed in the framework of *constraint classification*, introduced as an extension of standard classification by Har-Peled et al. (2002; 2003). The learning method proposed in this work constructs two training examples for each preference $\lambda_i \succ \lambda_j$, where the original N-dimensional training examples are mapped into a $(m \times N)$-dimensional space. The positive example copies the original training vector into the components $((i-1) \times N + 1) \dots (i \times N)$ and its negation into the components $((j-1) \times N + 1) \dots (j \times N)$ of a vector in the new space. The remaining entries are filled with 0, and the negative example has the same elements with reversed signs. In this $(m \times N)$-dimensional space, the learner tries to find a separating hyperplane. For classifying a new example e, the labels are ordered according to the response resulting from multiplying e with the i-th N-element section of the hyperplane vector.

Dekel et al. (2004) suggest a variation of boosting for learning a utility function in the form of a linear combination of a set of pre-defined base functions. They show that the resulting algorithm minimizes a generalized ranking error function that assumes a procedure for decomposing a preference graph into subgraphs, and defines the generalized error as the fraction of subgraphs that are misranked by the learned utility function.

2.4 Learning Preference Relations

It has already been noted that observed preferences are usually of the relational type, since utility scores are difficult to elicit. For example, it is very hard to ensure a consistent scale even if all utility evaluations are performed by the same user. Pyle (1999, p.16) claims that it is easier for a human to determine an order between several items if one makes pairwise comparisons between the individual items and then adds up the wins for each item, instead of trying to order the items right away.[2] The situation becomes even more problematic if utility scores are elicited from different users, which may not have a uniform scale of their scores (Cohen et al., 1999).

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. This technique essentially reduces the problem of learning preferences to a binary classification task. The learner is trained to predict for each pair of objects/labels which one is preferable. Pyle (1999) proposes a very

2. The aspect of being able to rank the available classifications for each example (as an intermediate version between predicting only a class value and providing a full probability distribution) is another interesting aspect of round robin binarization, which might be worth exploring in more depth.

similar technique called *pairwise ranking* in order to facilitate human decision-making in ranking problems. 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.

Note that the preference relation induced by a utility function is necessarily a total order, whereas the converse is not true: not every learned binary preference relation induces a total order that could be represented by a utility function. The point is that the learned preference relation does not necessarily have the typical properties of order relations, for example, it is not necessarily transitive. In fact, to obtain a ranking (total strict order) of the items, one has to apply a *ranking procedure* that accepts a binary preference relation as input and produces a ranking as output. The simplest approach is *ranking by scoring*: A score, such as the number of pairwise comparisons in which an item is preferred, is derived from the binary relation, and the items are then ordered according to their scores. For alternative (more complex) combination schemes see e.g. (Wu et al., 2004; Hüllermeier and Fürnkranz, 2004b).

For object ranking problems, the pairwise approach has been pursued in (Cohen et al., 1999). 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 (2003). 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. This approach is a natural extension of pairwise classification, i.e., the idea to solve a multi-class classification problem by learning separate theories for each pair of classes.

3. Label Ranking by Learning Pairwise Preferences

We consider a formal setting which can be considered as an extension of the conventional setting of classification learning. Roughly speaking, the former is obtained from the latter through replacing single class labels by complete label rankings: Instead of associating every instance x from an instance space $\mathcal{X}$ with one among a finite set of class labels $\mathcal{L} = \{\lambda_1 \ldots \lambda_m\}$, we now associate x with a complete ranking $\tau \in \mathcal{S}_m$ over these labels. More specifically, and again in analogy with the classification setting, every instance is associated with a *probability distribution* over the class of rankings (permutations) $\mathcal{S}_m$.[3] That is, for every instance x, there exists a probability distribution $\mathbb{P}(\cdot \,|\, x)$ such that, for every $\tau \in \mathcal{S}_m$, $\mathbb{P}(\tau \,|\, x)$ is the probability to observe the ranking τ as an output, given the instance x as an input.

The key idea of pairwise learning is well-known in the context of classification (Fürnkranz, 2002), where it allows one to transform a multi-class classification problem, i.e., a problem involving $m > 2$ classes $\mathcal{L} = \{\lambda_1 \ldots \lambda_m\}$, into a number of *binary* problems. To this

3. As one consequence of this assumption, note that one can observe 'clashes' in the training data (different label rankings for the same input).

end, a separate model (base learner) $\mathcal{M}_{ij}$ is trained for each *pair* of labels $(\lambda_i, \lambda_j) \in \mathcal{L}$, $1 \leq i < j \leq m$; 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, i.e., for any given example x, the model decides whether $\lambda_i \succ_x \lambda_j$ or $\lambda_j \succ_x \lambda_i$ holds. Thus, $\mathcal{M}_{ij}$ can be implemented as a binary classifier that outputs 1 if $\lambda_i \succ_x \lambda_j$ and 0 if $\lambda_j \succ_x \lambda_i$, i.e., which is intended to learn the mapping

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

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.

At classification time, a given instance x is submitted to all models $\mathcal{M}_{ij}$ and their predictions are combined into an overall prediction. In the simplest case, each prediction $\mathcal{M}_{ij}(x)$ is interpreted as a vote for a label. If $\mathcal{M}_{ij}(x) = 1$, this is counted as a vote for λ_i. Conversely, the prediction $\mathcal{M}_{ij}(x) = 0$ would be considered as a vote for λ_j. The label with the highest number of votes is proposed as a prediction. Ties can be broken in favor or prevalent classes, i.e., according to the class distribution in the classification setting.

Pairwise classification has been tried in the areas of statistics (Bradley and Terry, 1952; Friedman, 1996), neural networks (Knerr et al., 1990; 1992; Price et al., 1995; Lu and Ito, 1999), support vector machines (Schmidt and Gish, 1996; Hastie and Tibshirani, 1998; Kreßel, 1999; Hsu and Lin, 2002), 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 4.1).

The above procedure can be extended to the case of preference learning in a natural way (Fürnkranz and Hüllermeier, 2003). Again, a preference information of the form $\lambda_i \succ_x \lambda_j$ is turned into a training example (x, y) for the learner $\mathcal{M}_{ab}$, where $a = \min(i, j)$ and $b = \max(i, j)$. Moreover, $y = 1$ if $i < j$ and $y = 0$ otherwise. Figure 3 shows an illustration of this problem 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*).

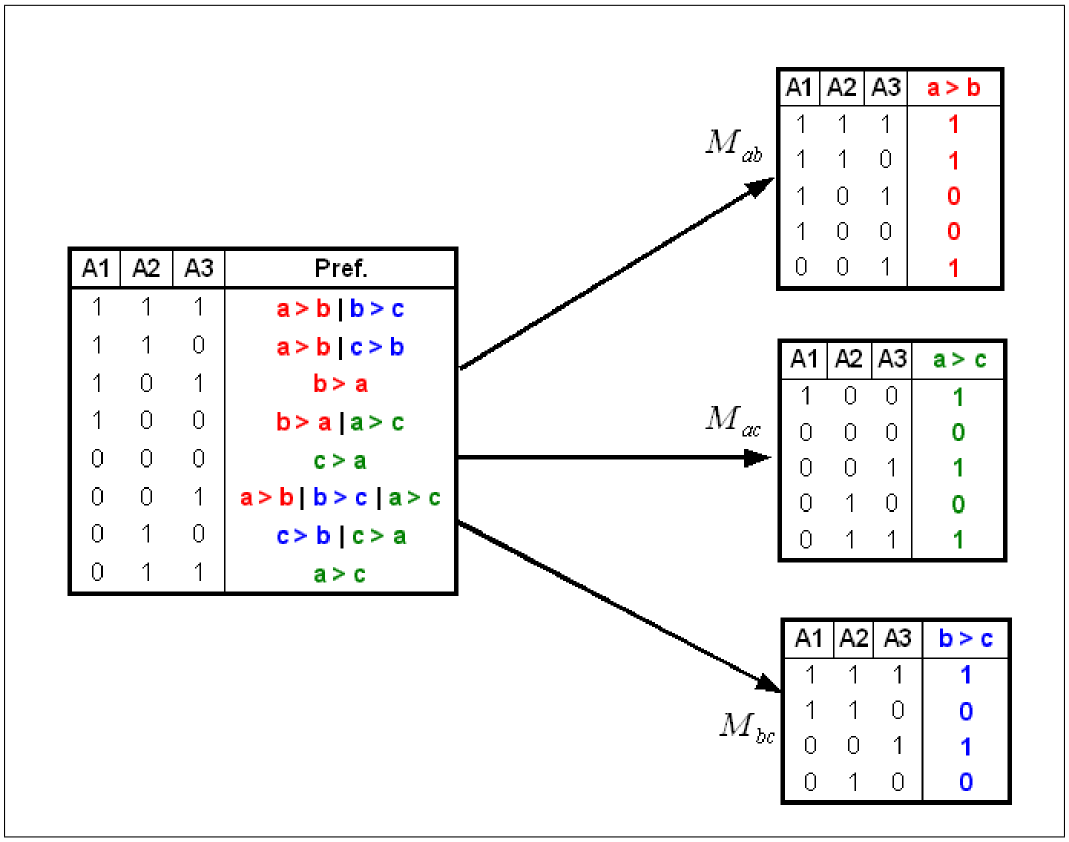
The mapping (1) can be realized by any binary classifier. Alternatively, one might of course employ a classifier that maps into the unit interval $[0, 1]$ instead of $\{0, 1\}$. The output of such a "soft" binary classifier can usually be interpreted as a probability or, more generally, a kind of confidence in the classification: the closer the output of $\mathcal{M}_{ab}$ to 1, the stronger the preference $\lambda_a \succ_x \lambda_b$ is supported.

A preference learner composed of an ensemble of soft binary classifiers assigns a *valued preference relation* $\mathcal{R}_x$ to every (query) instance $x \in \mathcal{X}$:

$$\mathcal{R}_x(\lambda_i, \lambda_j) = \begin{cases} \mathcal{M}_{ij}(x) & \text{if} \quad i < j \\ 1 - \mathcal{M}_{ij}(x) & \text{if} \quad i > j \end{cases} \tag{2}$$

for all $\lambda_i \neq \lambda_j \in \mathcal{L}$. Given such a preference relation $\mathcal{R}_x$ for an instance x, the next question is how to derive an associated ranking τ_x. As already discussed above, this question is non-trivial, since a relation $\mathcal{R}_x$ does not always suggest a unique ranking in an unequivocal

4. Rifkin and Klautau (2004) 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.



A1	A2	A3	Pref.
1	1	1	a > b | b > c
1	1	0	a > b | c > b
1	0	1	b > a
1	0	0	b > a | a > c
0	0	0	c > a
0	0	1	a > b | b > c | a > c
0	1	0	c > b | c > a
0	1	1	a > c

A1	A2	A3	a > b
1	1	1	1
1	1	0	1
1	0	1	0
1	0	0	0
0	0	1	1

A1	A2	A3	a > c
1	0	0	1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1

A1	A2	A3	b > c
1	1	1	1
1	1	0	0
0	0	1	1
0	1	0	0

Figure 3: Schematic illustration of a preference learning dataset (left) and its conversion into pairwise binary datasets (right)

way. 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 (Fodor and Roubens, 1994). In the context of pairwise classification and preference learning, several studies have empirically compared different ways of combining the predictions of individual classifiers (Wu et al., 2004; Allwein et al., 2000; Hüllermeier and Fürnkranz, 2004b; Fürnkranz, 2003).

The perhaps most simple approach is to make use of a straightforward extension of the aforementioned voting strategy for classification: The alternatives λ_i are evaluated by means of the sum of (weighted) votes

$$S(\lambda_i) = \sum_{\lambda_j \neq \lambda_i} \mathcal{R}_x(\lambda_i, \lambda_j) \tag{3}$$

and then ordered according to these evaluations, i.e.,

$$(\lambda_i \succeq \lambda_j) \Leftrightarrow (S(\lambda_i) \geq S(\lambda_j)). \tag{4}$$

This is a particular type of "ranking by scoring" strategy, where the scoring function is given by (3).[5] Even though this ranking procedure may appear rather ad hoc at first sight, we shall give a theoretical justification in Section 4.2, where it will be shown that ordering the labels according to (3) minimizes a reasonable loss function on rankings. In the experimental section, we shall also include a "binary" variant of RPC, called RPC-bin. Here, the idea is to map the "soft" prediction $\mathcal{M}_{ij}(x)$ of a learner to $\{0, 1\}$ before scoring the labels. In other words, the relation (2) is replaced by

$$\mathcal{R}_x(\lambda_i, \lambda_j) = \begin{cases} 1 & \text{if} \quad \mathcal{M}_{ij}(x) \geq 0.5 \\ 0 & \text{if} \quad \mathcal{M}_{ij}(x) < 0.5 \end{cases} \tag{5}$$

Even though (5) seems to come along with a loss of information, it turns out that this transformation might be useful under certain conditions. The point is that (5) can be considered as a kind of "reinforcement" of a prediction (e.g., it turns a 0.9 into a 1), which might be reasonable if the original predictions are already reliable enough (i.e., in the case of strong enough base learners); see (Hüllermeier and Fürnkranz, 2004b) for a detailed discussion of this issue.

In summary, our approach, referred to as *ranking by pairwise comparison* (RPC), consists of the following two steps:

- the derivation of a valued preference relation (2) by training an ensemble of (soft) binary classifiers, and
- the subsequent ranking of labels, using a ranking procedure such as (3–4).

We like to emphasize the *modularity* of RPC thus defined as a particular advantage of the approach. This modularity allows, for example, to adapt the ranking procedure in the second step to the problem at hand. In fact, as will be seen in Section 4.2, this allows to adjust RPC to minimize different loss functions on label rankings without the need for re-training the pairwise classifiers.

4. Formal Analysis

In this section, we will discuss several properties of RPC, starting with its computational complexity (Section 4.1). In Section 4.2, we will demonstrate that the same ensemble of pairwise classifiers can be used to minimize different ranking loss functions, including Spearman rank correlation and Kendall's tau. In Section 4.3, we discuss the relation of RPC to voting theory. Finally, Section 4.4 addresses limitations and potential improvements of the approach.

4.1 Complexity

As noted above, ranking by pairwise comparison is a straightforward generalization of pairwise aka one-against-one or round robin classification. In previous work, Fürnkranz (2002)

5. Strictly speaking, (3) does not necessarily define a ranking, as it does not exclude the case of indifference between labels. In such cases, a ranking can be enforced by any tie braking strategy.

analyzed the computational complexity of this approach and showed that, despite its complexity being quadratic in the number of classes, the algorithm is no slower than the conventional one-against-all technique. To understand this result, note that each training example is used m times (namely in each of the m binary problems) in the one-against-all case but only $m-1$ times in round robin (namely in those binary problems where its own class is paired with one of the other $m-1$ classes). Furthermore, the advantage of pairwise classification increases for computationally expensive (super-linear) learning algorithms. The reason is that expensive learning algorithms learn many small problems much faster than a few large problems. For details we refer to (Fürnkranz, 2002).

In this section, we will generalize these results for preference learning. In particular, we will show that this approach can be expected to be computationally more efficient than the approach of Har-Peled et al. (2002), which transforms the preference learning problem into a binary classification problem in a higher-dimensional space.

First, we will bound the number of training examples used by the pairwise approach. Recall that $|P_k|$ is the number of preferences that are associated with example x_k. We define d as the maximum of these values.

Theorem 1 *The total number of training examples constructed by RPC is bounded by nd, which is in turn bounded by $nm(m-1)/2$, i.e.,*

$$\sum_{k=1}^{n} |P_k| \leq nd \leq n\frac{m(m-1)}{2}$$

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 $\sum_{k=1}^{n} |P_k|$. As the number of preferences for each example is bounded from above by $d = \max_k |P_k|$, this number is no larger than dn, which in turn is bounded from above by the size of a complete set of preferences $nm(m-1)/2$. $\square$

For the particular case of classification, the above result can be specialized, as has been shown by Fürnkranz (2002):

Theorem 2 *For a classification problem, the total number of training examples is only linear in the number of classes.*

Proof: A single class label expands to $m-1$ preferences, because it is preferred over all other class labels. Therefore, $\sum_{k=1}^{n} |P_k| = (m-1)n$. $\square$

For comparison, the constraint classification approach of Har-Peled et al. (2002; 2003) converts each example into a set of examples, one positive and one negative example for each preference. Therefore, the original training data is transformed into a set of $2\sum_{k=1}^{n} |P_k|$ examples, which means that constraint classification constructs twice as many training examples as RPC.

Also note that the newly constructed examples are projected into a space that has m times as many attributes as the original space. However, Har-Peled et al. (2002; 2003) also note that when perceptrons are used as the base classifier, the algorithm can be implemented more efficiently in the form of a multi-output perceptron that is trained on each individual preference, quite similar to the approach of Crammer and Singer (2003a).

So far, we only considered the number of training examples, but not the complexity of the learner that runs on these examples. For an algorithm with a linear run-time complexity $\mathcal{O}(n)$ it follows immediately that the total run-time is $\mathcal{O}(dn)$, where d is the maximum (or average) number of preferences given for each training example.

Theorem 3 *For a base learner with complexity $\mathcal{O}(n^a)$, the total complexity of RPC is $\mathcal{O}(m^2n^a)$.*

Proof: RPC constructs one learning problem for each of the $m(m-1)/2$ possible preferences. In the worst case (a complete preference set), each problem may contain all n training examples, and the complexity for learning each problem is $\mathcal{O}(n^a)$. Thus, the total complexity is $(m(m-1)/2)\mathcal{O}(n^a) = \mathcal{O}(m^2n^a)$. $\square$

As discussed above, however, this bound is not tight. For example, one can show that for pairwise classification, the complexity is only linear in the number of classes $\mathcal{O}(mn^a)$ (Fürnkranz, 2002).

Theorem 4 *For a base learner with complexity $\mathcal{O}(n^a)$, the total complexity of constraint classification is $\mathcal{O}(m^{2a}n^a)$.*

Proof: Constraint classification constructs a single learning problem with, in the worst case, $2nm(m-1)/2 = \mathcal{O}(m^2n)$ training examples. If this problem is solved with a learner of complexity $\mathcal{O}(n^a)$, the total complexity is $\mathcal{O}(m^2n)^a = \mathcal{O}(m^{2a}n^a)$. $\square$

In summary, the overall complexity of pairwise constraint classification depends on the (maximum or 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 the technique proposed by Har-Peled et al. (2002), and can be considerably cheaper if the complexity of the base learner is super-linear (i.e., $a > 1$).

However, it should be noted that a possible disadvantage is the large number of classifiers that have to be stored. In principle, pairwise classification is more expensive than alternative approaches at classification time, because it has to query a quadratic number of classifiers, whereas one-against-all approaches or the multi-output perceptron advocated by Har-Peled et al. (2002; 2003) only need to compute a linear number of outputs. However, as binary classifiers resulting from RPC are constructed from fewer training examples and hence are typically less complex, it is possible to reduce the computational complexity at query time for certain base classifiers (such as support vector machines) with smart bookkeeping such that RPC is actually more efficient that one-against-all (Platt, 1999).

4.2 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), leading to different predictions. In any case, one might wonder whether the rankings predicted on the basis of (2) and (3) do have any kind of optimality property. An affirmative answer to this question will be given in this section.

4.2.1 Preliminaries

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

$$\mathbb{E}\,\left(\,D(y,\mathcal{M}(x))\,\right), \tag{6}$$

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 (e.g., the set $\mathcal{L}$ of classes in classification).[6]

4.2.2 Spearman rank correlation

An important and frequently applied similarity measure for rankings is the *Spearman rank correlation.* It was originally proposed as a nonparametric rank statistic by Spearman (1904) to measure the strength of the associations between two variables (Lehmann and D'Abrera, 1998). Formally, it is defined as follows:

$$1-\frac{6D(\tau,\tau')}{m(m^2-1)} \tag{7}$$

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

$$D(\tau',\tau)\overset{\text{df}}{=}\sum_{i=1}^{m}\left(\,\tau'(i)-\tau(i)\,\right)^2 \tag{8}$$

to the interval $[-1,1]$. As will now be shown, RPC is a risk minimizer with respect to (8) (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.,

$$\mathcal{R}_x(\lambda_i,\lambda_j)=\mathcal{M}_{ij}(x)=\mathbb{P}(\lambda_i\succ_x\lambda_j). \tag{9}$$

That is, if (9) holds, then RPC yields a risk minimizing prediction

$$\hat{\tau}_x=\arg\min_{\tau\in\mathcal{S}_m}\sum_{\tau'\in\mathcal{S}_m}D(\tau,\tau')\cdot\mathbb{P}(\tau'\,|\,x) \tag{10}$$

if $D(\cdot)$ is given by (8). Admittedly, (9) is a relatively strong assumption, as it requires the pairwise preference probabilities to be perfectly learnable. Yet, the result (10) 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

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

(9) 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.[7]

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

$$\sum_{i=1}^{m} (i - s_i)^2 \;\leq\; \sum_{i=1}^{m} (i - s_{\tau(i)})^2 \tag{11}$$

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 $\sum_{i=1}^m i\, s_{\tau(i)}$ depends on τ. This term is maximal for $\tau(i) = i$, because $s_i \leq s_j$ for $i < j$, and therefore $\max_{i=1\dots m} m s_i = m s_m$, $\max_{i=1\dots m-1} (m-1) s_i = (m-1) s_{m-1}$, 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. $\square$

Lemma 2 *Let $\mathbb{P}(\cdot \,|\, x)$ be a probability distribution over $\mathcal{S}_m$. Moreover, let*

$$s_i \overset{\text{df}}{=} m - \sum_{j \neq i} \mathbb{P}(\lambda_i \succ_x \lambda_j) \tag{12}$$

with

$$\mathbb{P}(\lambda_i \succ_x \lambda_j) = \sum_{\tau \,:\, \tau(i) < \tau(j)} \mathbb{P}(\tau \,|\, x). \tag{13}$$

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

7. Besides, note that so far the exact complexity of computing a Spearman-optimal aggregation for a set of rankings has been an open research problem (Dwork et al., 2001) and is now being solved as a special case of the result to be presented in this section. More precisely, the previously best-known technique for solving this aggregation problem is based on finding a minimum cost perfect matching in a bipartite graph (Dwork et al., 2001), which requires computational time of order $\mathcal{O}(m^3)$ (Ball and Derigs, 1983).

Proof: We have

$$
\begin{aligned}
s_i &= m - \sum_{j \neq i} \mathbb{P}(\lambda_i \succ_x \lambda_j) \\
&= 1 + \sum_{j \neq i} (1 - \mathbb{P}(\lambda_i \succ_x \lambda_j)) \\
&= 1 + \sum_{j \neq i} \mathbb{P}(\lambda_j \succ_x \lambda_i) \\
&= 1 + \sum_{j \neq i} \sum_{\tau \,:\, \tau(j) < \tau(i)} \mathbb{P}(\tau \,|\, x) \\
&= 1 + \sum_{\tau} \mathbb{P}(\tau \,|\, x) \sum_{j \neq i} \begin{cases} 1 & \text{if} \quad \tau(i) > \tau(j) \\ 0 & \text{if} \quad \tau(i) < \tau(j) \end{cases} \\
&= 1 + \sum_{\tau} \mathbb{P}(\tau \,|\, x)(\tau(i) - 1) \\
&= \sum_{\tau} \mathbb{P}(\tau \,|\, x) \, \tau(i)
\end{aligned}
$$

□

Note that $s_i \leq s_j$ is equivalent to $S(\lambda_i) \geq S(\lambda_j)$ (as defined in (3)) under the assumption (9). 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 5 *The expected distance*

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

becomes minimal by choosing τ' such that $\tau'(i) \leq \tau'(j)$ whenever $s_i \leq s_j$, with s_i given by (12).

Proof: We have

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

In the last equation, the mid-term on the right-hand side becomes 0 according to Lemma 2. Moreover, the last term obviously simplifies to $(s_i - \tau'(i))$, 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 1. □

4.2.3 Kendall's tau

The above result shows that our approach to label ranking in the form presented in Section 3 is particularly tailored to (8) 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.[8] 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* Kendall (1955) 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

$$D(\tau', \tau) \overset{\text{df}}{=} \#\{(i, j) \mid i < j, \tau(i) > \tau(j) \wedge \tau'(i) < \tau'(j)\} \tag{14}$$

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 $[-1, +1]$.

8. Even though it is true that not every loss function can be minimized, as will be shown in Section 4.4.2.

Now, for every ranking τ',

$$\begin{aligned}\mathbb{E}(D(\tau',\tau) \mid x) &= \sum_{\tau \in \mathcal{S}_m} \mathbb{P}(\tau) \times D(\tau',\tau) \qquad (15)\\ &= \sum_{\tau \in \mathcal{S}_m} \mathbb{P}(\tau \mid x) \times \sum_{i<j \,|\, \tau'(i)<\tau'(j)} \begin{cases} 1 & \text{if} \quad \tau(i) > \tau(j) \\ 0 & \text{if} \quad \tau(i) < \tau(j) \end{cases} \\ &= \sum_{i<j \,|\, \tau'(i)<\tau'(j)} \sum_{\tau \in \mathcal{S}_m} \mathbb{P}(\tau \mid x) \times \begin{cases} 1 & \text{if} \quad \tau(i) > \tau(j) \\ 0 & \text{if} \quad \tau(i) < \tau(j) \end{cases} \\ &= \sum_{i<j \,|\, \tau'(i)<\tau'(j)} \mathbb{P}(\lambda_i \succ_x \lambda_j)\end{aligned}$$

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 (14) 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 (15) is formally equivalent to solving the graph-theoretical *feedback arc set* problem (for *weighted* tournaments) which is known to be NP complete Alon (2000). 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.

4.3 Connections with Voting Theory

It is worth mentioning that RPC is closely related to the so-called *Borda-count*, a voting rule that is well-known in social choice theory (Brams and Fishburn, 2002): 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:

$$\hat{\tau} = \arg\min_{\tau \in \mathcal{S}_m} \sum_{i=1}^{n} D(\tau, \tau_i). \qquad (16)$$

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 (1998).

A 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 (14) as a distance measure $D(\cdot)$ in (16), $\hat{\tau}$ is also called the *Kemeny-optimal* ranking. Kendall's tau is intuitively quite appealing and Kemeny-optimal rankings have several nice properties. However, one drawback of using Kendall' tau instead of rank correlation as a distance measure in (16) is a loss of computational efficiency. In fact, the computation of Kemeny-optimal rankings is known to be NP-hard (Bartholdi et al., 1989).

4.4 Properties and Consequences of Pairwise Decomposition

Our pairwise learning scheme, in the basic version presented so far, decomposes preference information into pairwise relations in a first step. Intuitively, this decomposition should come along with a certain loss of information. Besides, one may wonder whether learning pairwise preferences independently of each other may perhaps neglect certain interdependencies between such preferences. This section is meant to briefly touch on corresponding limitations and potential improvements of our approach to learning by pairwise comparison.

4.4.1 Interdependencies between Pairwise Models

Regarding potential interdependencies between pairwise preferences, an obvious candidate is *transitivity*, which is one of the most important properties in preference modeling. Indeed, the pairwise preferences induced by a *single* ranking are obviously transitive. What is less clear, however, is 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 $\mathcal{S}_m$. Such a distribution induces a unique probability distribution for pairwise preferences via

$$p_{ij} = \mathbb{P}(\lambda_i \succ \lambda_j) = \sum_{\tau \in \mathcal{S}_m \,:\, \tau(i) < \tau(j)} \mathbb{P}(\tau). \tag{17}$$

An interesting finding is that the pairwise preferences (17) do indeed satisfy a form of transitivity, albeit a relatively weak one:

$$\forall\, i, j, k \in \{1 \ldots m\} \,:\, p_{ik} \geq p_{ij} + p_{jk} - 1 \tag{18}$$

More formally, we can prove the following theorem.

Theorem 6 *Consider any probability distribution on the set of rankings $\mathcal{S}_m$. The pairwise preferences induced by this distribution via (17) satisfy (18).*

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 $S_{ijk} = \{\tau \in \mathcal{S}_m \,|\, \tau(i) < \tau(j) < \tau(k)\}$ 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 (17). 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 $p_{ik} \geq p_{ij}$ (fourth entry) and (18) 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

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

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

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

which is non-negative if and only if $p_{ik} \leq p_{ij} + p_{jk}$. This latter inequality is equivalent to $p_{kj} \geq p_{kj} + p_{ji} - 1$, where $p_{kj} = 1 - p_{jk}$, so the transitivity property (18) now holds for the reciprocal probabilities. In a similar way one verifies that (18) 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 (18). $\square$

It is interesting to note that (18) 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 (Schweizer and Sklar, 1983) and have been studied intensively in fuzzy set theory in recent years (Klement et al., 2002). A binary relation $\mathcal{R} \subset A \times A$ is called $\top$-transitive if it satisfies $\mathcal{R}(a, c) \geq \top(\mathcal{R}(a, b), \mathcal{R}(b, c))$ for all $a, b, c \in A$. Therefore, what the condition (18) expresses is just $\top$-transitivity with respect to the Lukasiewicz t-norm which is defined by $\top(x, y) = \max(x + y - 1, 0)$.

The above result has shown that, roughly speaking, if p_{ij} and p_{jk} are large, then p_{ik} must not be too small, and vice versa, if p_{ij} and p_{jk} are small, then p_{ik} must not be too large; more precisely,

$$p_{ij} + p_{jk} - 1 \;\leq\; p_{ik} \;\leq\; p_{ij} + p_{jk}. \tag{19}$$

These constraints are relatively weak. In particular, only one of the two constraints can really grab, since the left one becomes trivial if $p_{ij} + p_{jk} \leq 1$ and the right one if $p_{ij} + p_{jk} \geq 1$. Nevertheless, the predictions obtained by an ensemble of pairwise learners should actually satisfy (19). 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 (19) is not very high. Still, forcing (19) to hold is a potential point of improvement, and there seem to exist different approaches in which this could be done. A simple idea, for example, it to replace the original ensemble of pairwise predictions by its $\top$-transitive closure Naessens et al. (2002), where $\top$ is the aforementioned Lukasiewicz t-norm. Elaborating on ideas of that kind in more detail is an important topic of future work.

4.4.2 Inherent Loss of Information

Even if complete rankings are available for training, the principle of pairwise learning dictates to split these rankings into pairwise preferences, which are then submitted to the base learners $\mathcal{M}_{ij}$. Correspondingly, a probability measure over complete rankings (the output space in the label ranking problem) is replaced by pairwise probabilities of the form $\mathbb{P}(\lambda_i \succ \lambda_j)$ according to (17). As already mentioned before, this decomposition may come along with a certain loss of information. Indeed, note that the transformation (17) is irreversible: Given the probabilities $\mathbb{P}(\lambda_i \succ \lambda_j)$ for pairwise preferences, it is not possible to recover the underlying distribution $\mathbb{P}(\cdot)$ on $\mathcal{S}_m$. To illustrate, consider the following distributions $\mathbb{P}(\cdot \,|\, x) \neq \mathbb{P}'(\cdot \,|\, x)$ for an instance x:

$$\mathbb{P}(\tau \,|\, x) = \begin{cases} 1/2 & \text{if } \tau = (1\,2\,3\,4) \\ 1/2 & \text{if } \tau = (4\,3\,2\,1) \\ 0 & \text{otherwise} \end{cases}, \quad \mathbb{P}'(\tau \,|\, x) = \begin{cases} 1/2 & \text{if } \tau = (1\,3\,2\,4) \\ 1/2 & \text{if } \tau = (4\,2\,3\,1) \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

¿From both distributions, one derives the same probabilities $\mathbb{P}(\lambda_i \succ_x \lambda_j)$ for pairwise preferences:

$$\mathcal{R}_x = \begin{bmatrix} - & 1/2 & 1/2 & 1/2 \\ 1/2 & - & 1/2 & 1/2 \\ 1/2 & 1/2 & - & 1/2 \\ 1/2 & 1/2 & 1/2 & - \end{bmatrix}$$

Consequently, RPC cannot distinguish between the original distributions on $\mathcal{S}_m$. As one consequence of this information loss, we note that RPC cannot minimize every loss function. For example, consider the simple 0/1–loss which is commonly employed in classification: $D(y, \hat{y}) = 0$ for $y = \hat{y}$ and $= 1$ otherwise. If this loss function is used, the optimal (Bayes) prediction for a specific instance x is obviously given by the most probable outcome y. In the classification setting, for example, where $\mathcal{Y} = \mathcal{L}$, this estimate is the class with maximum posterior probability $\mathbb{P}(\lambda_i \,|\, x)$. A straightforward generalization of this principle to the ranking setting, where $\mathcal{Y}$ is the class of rankings over $\mathcal{L}$, leads to the prediction

$$\hat{\tau}_x \in \arg\max_{\tau \in \mathcal{S}_m} \mathbb{P}(\tau \,|\, x), \tag{21}$$

where $\mathbb{P}(\tau \,|\, x)$ is the conditional probability of a ranking (permutation) given an instance x. Obviously, (21) leads to different predictions for the two distributions $\mathbb{P}(\cdot \,|\, x)$ and $\mathbb{P}'(\cdot \,|\, x)$ in (20), which means that a risk minimizing prediction cannot be delivered by RPC.

Even though the above example shows that pairwise decomposition surely causes a loss of information, it is not obvious how bad this loss actually is, for example with respect to risk minimization. The fact that the 0/1–loss cannot be minimized is indeed tolerable, since this is definitely not a reasonable distance measure for rankings. The general question which loss functions can be minimized by RPC and which cannot is still an open problem.

5. Empirical Evaluation

The experimental evaluation presented in this section compares the pairwise ranking (RPC) with the constraint classification (CC) approach in terms of accuracy and computational efficiency. We considered three different scenarios, which can be roughly categorized as real-world, semi-synthetic, and synthetic, in order to provide a comprehensive analysis under varying conditions. The real-world scenario includes bioinformatics datasets of quite different complexity and is generally rather challenging. In order to generate the semi-synthetic data, we replaced single class labels in a variety of standard multiclass benchmark datasets with complete rankings. For the synthetic experimental data, we replicated a setting proposed by Fürnkranz and Hüllermeier (2003) which operates in the context of expected utility theory and allows us to control the complexity of the problem by means of adjusting the input dimension and the label set size.

5.1 Experimental Data

5.1.1 Real-world Data

This 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, consisting of 2465 genes.[9] Every gene was 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 a sequence of real-valued measurements, each of which represents the expression level of that gene measured at a particular time point. Converting the expression levels into ranks, i.e., ordering the time points (= labels) according to the associated expression values and using the Spearman correlation as a similarity measure between profiles was motivated in (Balasubramaniyan et al., 2005).[10]

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.

9. This data is publicly available at `http://www1.cs.columbia.edu/compbio/exp-phylo`
10. This transformation can be motivated from both a biological as well as data analysis point of view.

dataset	#examples	#classes	#features
iris	150	3	4
wine	178	3	13
glass	214	6	9
vowel	528	10	10
vehicle	846	4	18

Table 2: Dataset statistics

It is worth mentioning that these experiments involve different numbers of measurements, ranging from 4 to 11.[11] 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. Each of the datasets was randomly split into a training and a test set comprising 500 and 1965 instances, respectively. In compliance with (Balasubramaniyan et al., 2005), we measured accuracy in terms of the Spearman rank correlation coefficient (see Section 4.2).

5.1.2 Semi-Synthetic Data

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 (Blake and Merz, 1998) and the Statlog collection (Michie et al., 1994) were included in the experimental evaluation: iris, wine, glass, vowel, vehicle (a summary of dataset properties is given in Table 2). This collection describes a subset of the datasets tested in a recent experimental study on multiclass support vector machines (Hsu and Lin, 2002). Containing at least 2,000 examples, the remaining five datasets had to be excluded due to computational efficiency issues.

For each of these datasets, a corresponding ranking dataset was generated in the following manner: We trained a naive Bayes classifier[12] 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.

5.1.3 Synthetic Data

We replicated a setting proposed by Fürnkranz and Hüllermeier (2003) which operates in the context of expected utility theory: An expected utility maximizing agent is given a set

11. We excluded three additional subproblems with more measurements due to the prohibitive computational demands of the constraint classification approach.
12. We employed the implementation for naive Bayes classification on numerical datasets (`NaiveBayesSimple`) contained in the Weka machine learning package (Witten and Frank, 2000).

$\{\lambda_1, \dots, \lambda_m\}$ of alternative actions to choose from. The agent faces a problem of *decision under uncertainty* where alternative λ_i yields a utility value denoted by the matrix entry $U_{ij} \in \mathbb{R}$ if the world is in state $\omega_j \in \Omega = \{\omega_1, \dots, \omega_N\}$. The probability of state ω_j is denoted by the jth component p_j of the probability vector $p = (p_1, \dots, p_N)^\top$ and therefore the expected utility of alternative λ_i evaluates to

$$\mathbb{E}(\lambda_i) = \sum_{j=1}^{N} p_j \, U_{ij}. \tag{22}$$

Expected utility theory justifies (22) as a criterion for ordering actions, hence, giving rise to a natural order over the set of alternative actions: We assume the set of alternatives to be in decreasing order with respect to expected utility in the following. Let us assume the probability vector $p = (p_1, \dots, p_N)^\top$ to be the feature vector of a ranking example where the number of alternatives c and the set of world states Ω are fixed and the $m \times N$ utility matrix U has independently and uniformly distributed entries $U_{ij} \in [0, 1]$. Then, for a given probability vector p the above-defined decision-theoretic scenario gives rise to an order over the set of alternative actions. Now, a set of d feature vectors are independently drawn from a uniform distribution over $\{p \in \mathbb{R}^N \mid p_1 + \dots + p_N = 1,\ p_i \geq 0 \text{ for } i = 1, \dots, N\}$ and assigned to the corresponding permutations in order to generate a ranking dataset. Note that this setting corresponds to a noise-free scenario in the constraint classification framework (with linear kernels) since for a given feature vector p an alternative way of expressing the corresponding ranking is $\text{argsort}_{\lambda_1, \dots, \lambda_m} \langle \bar{u}_i^\top, p \rangle$ (with $\bar{u}_i$ denoting the ith row vector of U).

5.2 Experimental Setup

We used a kernelized version of the perceptron algorithm as the binary base learner in all experiments. In order to obtain more robust binary classifiers, we averaged the weight vectors generated by the kernel perceptron algorithm over 10 random permutations of the given training sets. This technique has also been proposed for extending the Bayes-point approach to large-scale learning problems (Herbrich and Graepel, 2001) and can be viewed as an approximate method to compute the center of the so-called version space[13]: The weight vectors generated by the perceptron algorithm on random permutations of the training set provide (pseudo) random samples from the version space and, thus, their average approximates the center of mass of the convex version space. Moreover, in the case of pairwise label ranking with soft voting, we adopted a common approach from the field of support vector learning to convert real-valued scores (classification scores before thresholding) into (pseudo-)probabilities using a logistic regression technique (Platt, 1999). Soft voting is computationally much more expensive than binary voting since computing the probabilities requires solving an optimization problem and makes use of an internal 3-fold cross-validation procedure for robustness.

In the experiments, the actual true rankings $\tau \in S_m$ on the test sets were compared to the corresponding predicted rankings $\tau' \in S_m$. Recall that we denote by $\tau(i)$ and $\tau'(i)$,

13. The version space is the set of all hypothesis consistent with a given training set.

respectively, the rank of a particular label λ_i in the given complete ranking. We computed the average accuracy for each of the approaches with respect to the following standard evaluation measures on complete rankings, that we already introduced in Section 4.2 and recall here for convenience:
Spearman Rank Correlation calculates the sum of squared rank distances and is normalized such that it evaluates to -1 for reversed and to $+1$ for identical rankings. Formally, it is defined as follows:

$$(\tau, \tau') \mapsto 1 - \frac{6\sum_{i=1}^{m}(\tau(i) - \tau'(i))^2}{m(m^2-1)}. \tag{23}$$

As shown in Section 4.2, RPC with ranking by weighted voting is particularly tailored to this evaluation measure since it yields a risk minimizing prediction of the corresponding loss function.

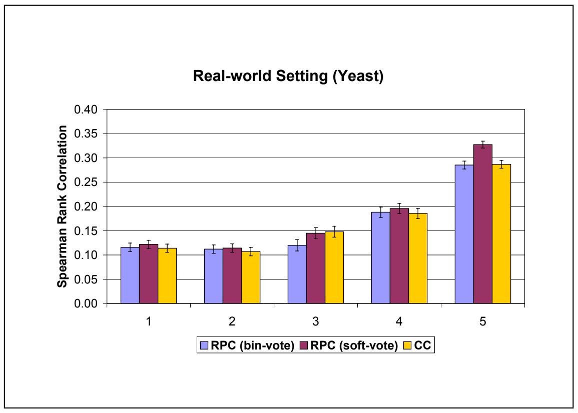
Kendall tau calculates the number of pairwise rank inversions on labels to measure the ordinal correlation of two rankings (Kendall, 1955):

$$(\tau, \tau') \mapsto 1 - \frac{4\left|\{(i,j) \mid i < j,\, \tau(i) < \tau(j) \,\wedge\, \tau'(i) > \tau'(j)\}\right|}{m(m-1)}. \tag{24}$$

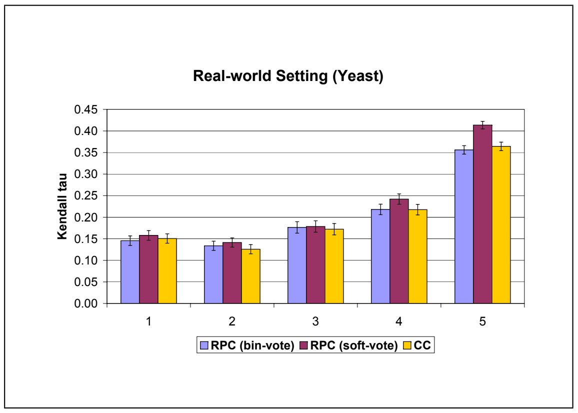
Recall from Section 4.2 that RPC can also be adjusted to Kendall tau as a loss function, simply by changing the ranking procedure. For computational reasons, however, we *did not use* this more complex procedure. Instead, we again resorted to the much more efficient voting approach (3–4). Interestingly, the constraint classification approach is instead more focused on the Kendall tau measure: Minimization of the 0/1-loss on the expanded set of (binary) classification examples yields an implicit minimization of the empirical Kendall tau statistic of the label ranking function on the training set. Thus, one may expect that constraint classification is slightly privileged by the Kendall tau measure while RPC is favored by Spearman rank correlation. One should also note, however, that all distance (similarity) measures on rankings are of course more or less closely related. For example, it has recently been shown in Coppersmith et al. (2006) that optimizing rank correlation yields a 5-approximation to the ranking which is optimal for the Kendall measure.

Note that for both measures we consider a normalization such that a higher degree of similarity results in a higher score. However, a straightforward transformation integrates these measures into the common loss/risk-framework.

In order to conduct a fair experimental study, we performed model selection on the real-world and semi-synthetic datasets with respect to the kernel parameters separately for each of the approaches. Note that in compliance with related studies in classification learning, the kernel parameters for pairwise ranking were selected in a global manner such that they are the same for all binary classifiers. We considered linear kernels with a penalty parameter $C \in \{2^{-4}, 2^{-3}, \ldots, 2^{10}\}$. For each choice of the penalty parameter C, we estimated the generalization accuracy of the label ranking function with respect to the particular measure considered in the given experiment by conducting 10-fold cross-validation on the training set. Then, the label ranking function was trained on the whole training set using the penalty parameter with the highest estimated accuracy. The actual generalization accuracy with respect to the rank correlation and Kendall tau was estimated using the 10-fold



(a) Rank Correlation



(b) Kendall Tau

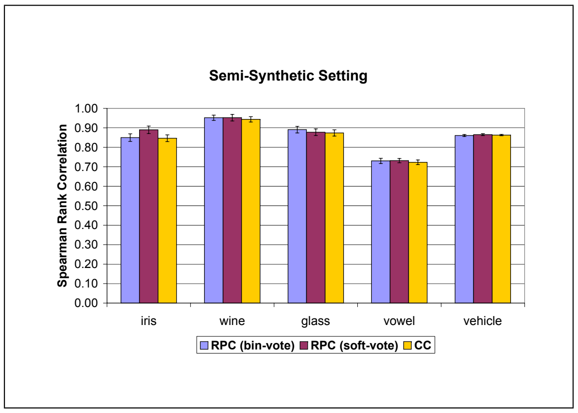
Figure 4: Real-world Setting (Yeast)

approach	iris		wine		glass		vowel		vehicle	
RPC (bin-vote)	12.2	(1.0)	17.1	(1.0)	157.8	(1.0)	4977.2	(1.0)	1956.7	(1.0)
RPC (soft-vote)	45.5	(3.7)	64.0	(3.7)	498.5	(3.2)	14167.3	(2.8)	6201.0	(3.2)
CC	36.9	(3.0)	62.0	(3.6)	1756.8	(11.1)	281194.8	(56.5)	6799.1	(3.5)

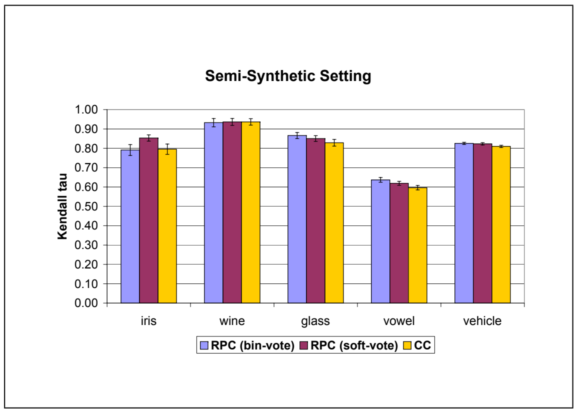
Table 3: Overall experimental running times in seconds (normalized results are given in brackets)

cross-validation technique again. Hence, the experimental setup consists of a nested two-level architecture where 10-fold cross-validation forms an essential building block on each level: The inner level estimation is necessary for selecting hyperparameters and the outer estimation for measuring accuracy.

The synthetic data setting was incorporated into the experimental setup to conduct controlled experiments under various training conditions. We varied both the number of alternatives ($m \in \{5, 10, 15, 20\}$) with the number of training examples being fixed to $n = 1000$ and the number of training examples ($n \in \{100, 250, 500, 750, 1000\}$) with the number of alternatives being fixed to $m = 10$. The number of test examples was fixed to 1000 in all cases. For each of the configurations, we generated 10 different pairs of training and test sets, each pair originating from a different randomly chosen utility matrix U. Moreover, as linear pairwise decision boundaries are optimal in this setting (see Equation (22)) we refrained from using a kernel parameter selection technique and employed a linear kernel (with $C = 1000$) for the pairwise ranking and the constraint classification techniques in all experiments.



(a) Rank Correlation



(b) Kendall Tau

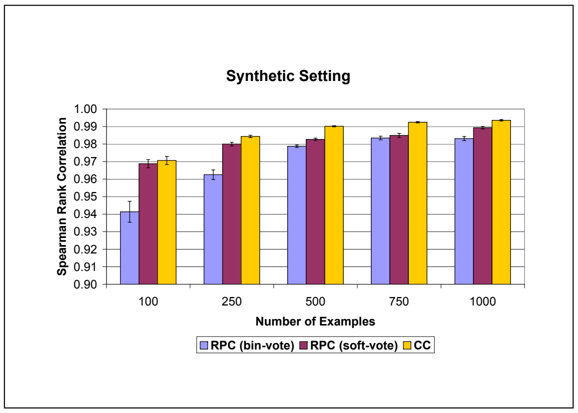
Figure 5: Semi-Synthetic Setting

5.3 Experimental Results

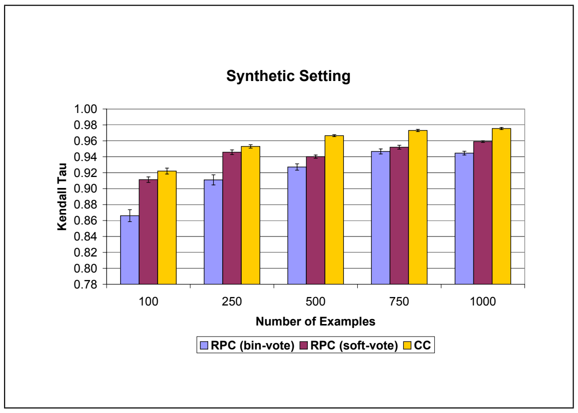
5.3.1 Complete Preference Information

Real-world Data: The experimental results of the pairwise ranking approach (RPC) with binary voting (bin-vote) and soft voting (soft-vote) in comparison to the constraint classification approach (CC) are depicted in Figure 4 (cf. also Tables 4 and 5 in the Appendix). Except for one experimental configuration, the ranking by pairwise comparison with soft voting always achieves the best accuracy. Moreover, there is no substantial difference between pairwise ranking with binary voting and the constraint classification approach. This scenario forms a particularly challenging test bed (as indicated by the absolute level of accuracy) and experimentally underpins a finding from (Hüllermeier and Fürnkranz, 2004a) claiming that binary voting is expected to be superior to soft voting in the high accuracy regime while the reverse holds for more challenging problems. As mentioned previously, binary voting can be interpreted as a reinforcement of soft predictions which is a reasonable technique if these predictions are sufficiently accurate.

Semi-Synthetic Data: Figure 5 shows experimental results of the pairwise ranking approach (RPC) with binary voting (bin-vote) and soft voting (soft-vote) in comparison to the constraint classification approach (CC) (cf. also Tables 6 and 9 in the Appendix). Moreover, Table 3 shows overall running times (in seconds) for the considered approaches. As indicated by the normalized experimental running times, the soft voting approach requires additional computational time by an average factor of approximately 3.3 compared to its binary voting counterpart in pairwise ranking (for converting output scores into probabilities). The constraint classification approach becomes increasingly more inefficient with respect to computational time with the number of possible labels growing. In particular, constraint classification is approximately 50 times slower than pairwise ranking with binary voting for the vowel dataset where the number of different classes amounts to 10. As shown in Theorem 4, pairwise expansion of constraints yields an algorithm with $\mathcal{O}(m^{2a}n^{a})$ computational training complexity, thus limiting the range of application to a small number of alternatives in the case of a superlinear base learning algorithm.

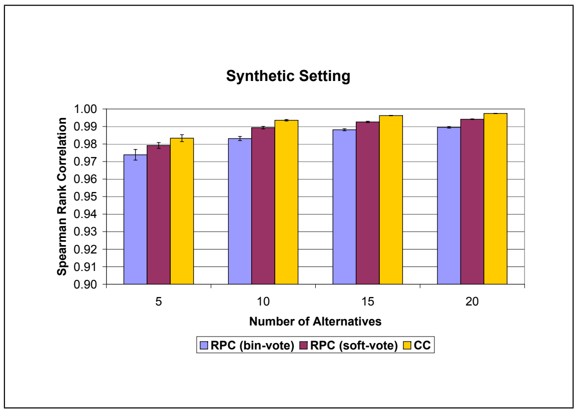


(a) Rank Correlation

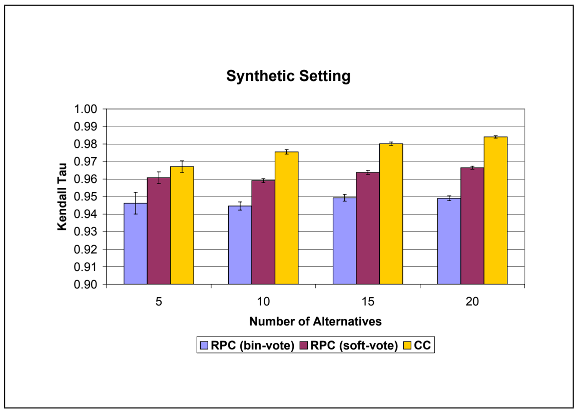


(b) Kendall tau

Figure 6: Synthetic Setting with varying numbers of training examples



(a) Rank correlation



(b) Kendall tau

Figure 7: Synthetic Setting with varying numbers of alternatives

Synthetic Data: In order to provide a more detailed empirical analysis, we conducted a series of controlled experiments with varying numbers of alternatives and training examples using the synthetic setting in Section 5.1. As expected, the accuracy of all approaches measured by the Spearman rank correlation coefficient and the Kendall tau increases in the number of training examples (see Figure 6). Moreover, there is a constant ranking among the different techniques with the constraint classification approach always performing best (as expected, the advantage of constraint classification is stronger for the Kendall tau than for rank correlation measure). An analogous observation can be made in the case of a varying number of alternatives with the number of training examples being fixed (see Figure 7). This pattern is not surprising since the considered synthetic setting clearly favors this technique as mentioned before: Constraint classification is based on precisely the hypothesis class which has been used for generating the datasets. Nevertheless, the difference in generalization accuracy between RPC with soft-voting and constraint classification turns

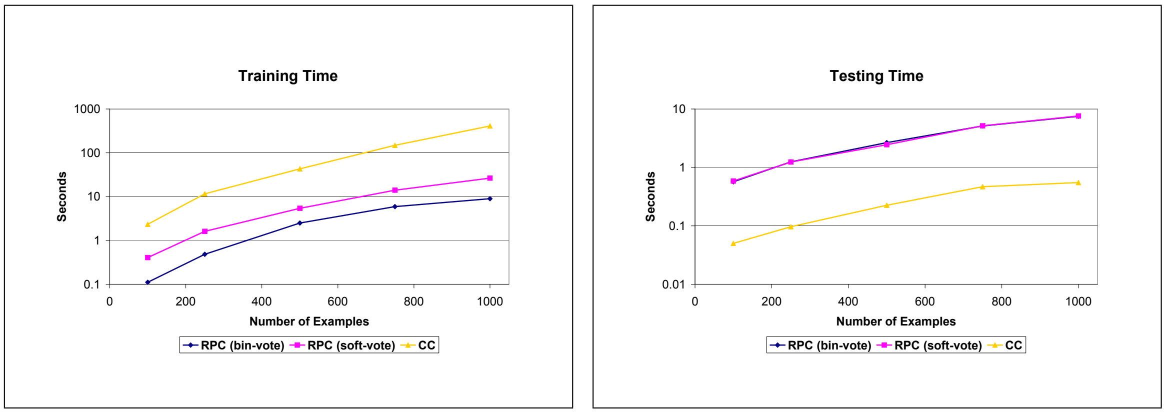


Figure 8: Training and testing times for the synthetic setting with varying numbers of training examples (plotted on a log-scale)

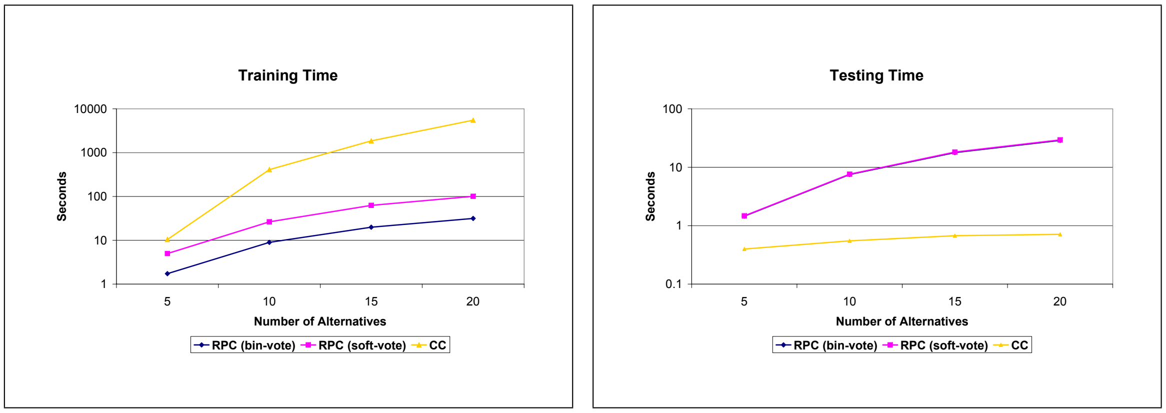
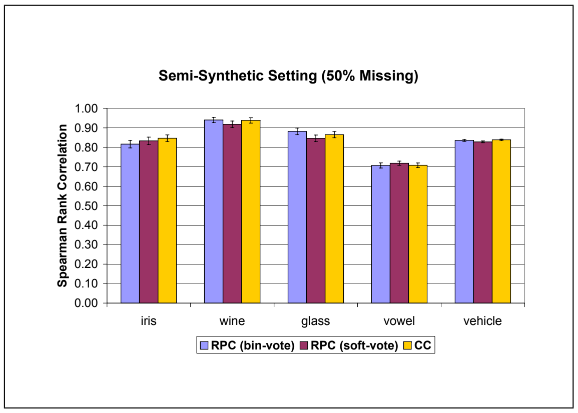


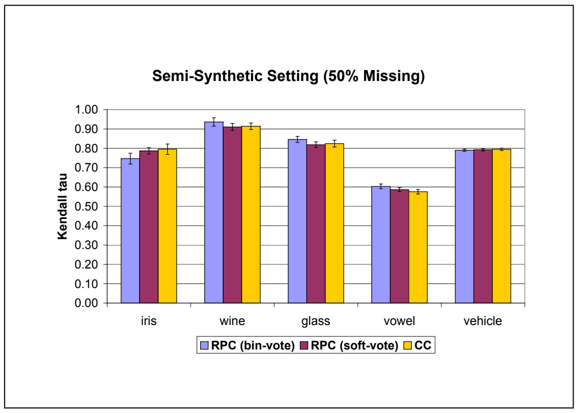
Figure 9: Training and testing times for the synthetic setting with varying numbers of alternatives (plotted on a log-scale)

out to be limited, thus, it still may become the favorable technique in this setting if aspects like training time are considered a major design issue for the preference learning system.

In terms of the computational complexity required for training, pairwise ranking substantially outperforms constraint classification (see Figures 8 and 9). The soft voting technique requires additional computational time by an average factor of approximately 3.0 compared to its binary voting counterpart in pairwise ranking. Conversely, in terms of testing complexity, the constraint classification approach substantially outperforms the pairwise ranking approaches, thus, complementing our theoretical analysis which pointed out that the number of binary classifiers to be evaluated scales linearly in the number of alternatives in the case of constraint classification and quadratically in the case of pairwise ranking.

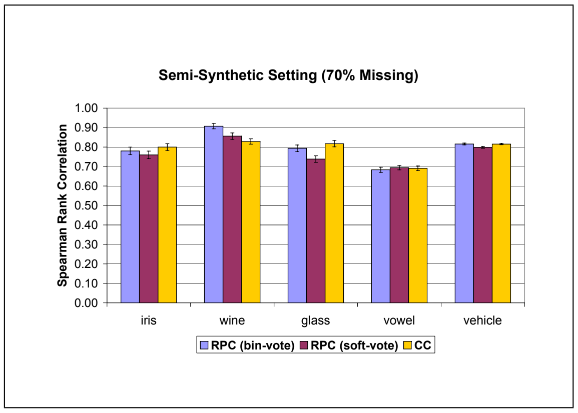


(a) Rank correlation

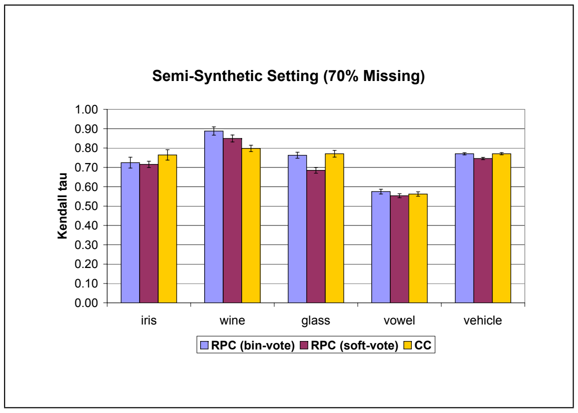


(b) Kendall tau

Figure 10: Semi-Synthetic Setting with 50% missing labels



(a) Rank correlation



(b) Kendall tau

Figure 11: Semi-Synthetic Setting 70% missing labels

5.3.2 Incomplete Preference Information

In Section 5.3.1, 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.

In the following, we will model missing preferences by assuming that a complete ranking is available for a subset of possible labels, i.e., when only partial rankings are submitted to the learning algorithms. More precisely, we randomly deleted a fraction of 50% and 70% respectively of all the labels given in the rankings of the training sets, such that an original ranking $\lambda_1 \succ \lambda_2 \succ \lambda_3 \succ \lambda_4 \succ \lambda_5$ may be reduced to $\lambda_1 \succ \lambda_3 \succ \lambda_4$, and hence, pairwise preferences were generated only from the partial rankings. As we assumed a uniform distribution over the entire set of labels present in the dataset for the process of deleting labels, in general, the resulting datasets consisted of examples associated with partial rankings of varying size.

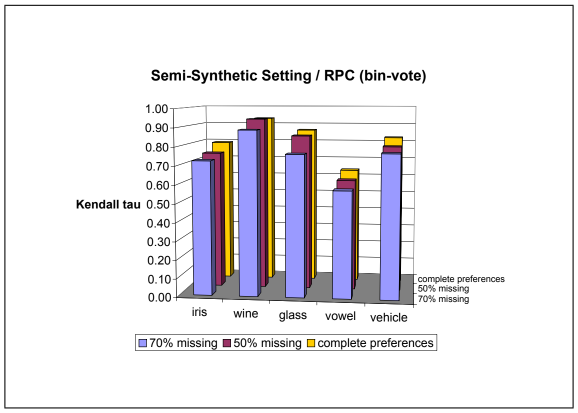


Figure 12: Semi-Synthetic Setting with RPC and 0%, 50% and 70% missing labels (Evaluation measure: Kendall tau)

A modification of the evaluation measures used in the inner cross-validation estimation for selecting hyperparameters was necessary since the predicted *complete* rankings had to be evaluated on the *partial* rankings in the cross-validation test sets. To this end, we used the following adaptation: Given a partial ranking (e.g., $\lambda_1 \succ \lambda_3 \succ \lambda_4$) and a complete ranking (e.g, $\lambda_1 \succ \lambda_2 \succ \lambda_3 \succ \lambda_4 \succ \lambda_5$), all the labels not present in the partial ranking are deleted from the complete ranking ($(\lambda_1 \succ \lambda_2 \succ \lambda_3 \succ \lambda_4 \succ \lambda_5) \mapsto (\lambda_1 \succ \lambda_4 \succ \lambda_3)$) and then the specific measure of accuracy is evaluated on the normalized partial rankings, i.e., on the ranking where the maximal rank corresponds to the number of remaining labels (in our example, the normalized true ranking is $\lambda_1 \succ \lambda_2 \succ \lambda_3$ and the normalized predicted ranking is $\lambda_1 \succ \lambda_3 \succ \lambda_2$). Note that the number of possible labels m necessary for evaluating the rank correlation and the Kendall tau, respectively, has to be substituted by the effective number of labels for the given normalized rankings (in the aforementioned example, $m = 3$).

Figures 10 and 11 visualize the experimental result for a fraction of 50% and 70%, respectively, being randomly deleted from the dataset (cf. also Tables 7, 8, 10 and 11 in the Appendix). As expected, the accuracy measured by the Spearman rank correlation and the Kendall tau of the respective label ranking functions decreases with the amount of incomplete preference information increasing. However, both RPC with binary- and soft-voting and constraint classification can deal with the more realistic case of limited preference information remarkably well and still achieve a high level of accuracy (see Figure 12 for an exemplary visualization for RPC-bin with Kendall tau). Furthermore, none of the considered approaches substantially outperforms the others.

6. Related Work

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 (Hsu and Lin, 2002, and references therein).

We refer to (Fürnkranz, 2002, Section 8) for a brief survey of work on pairwise classification, and its relation to other learning class binarization techniques, most notably the general framework of Allwein et al. (2000).

There has also been some recent work on label ranking algorithms for multi-label problems. For example, Crammer and Singer (2003b) 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. ¿From a more theoretical perspective, Usunier et al. (2006) analyze the generalization properties of binary classifiers trained on interdependent data for certain types of structured learning problems such as bipartite ranking. We are aware of only one work that actually uses a *complete* ranking of the available labels for each example for training or evaluation: Brazdil et al. (2003) investigate the meta-learning task of ranking learning algorithms according to their suitability for a new dataset, based on the characteristics of this dataset.

Depending on the underlying utility scale one can distinguish between learning a numerical function and learning a function that maps into an ordinal (ordered categorical) scale. These two cases involve, respectively, a problem of standard regression and ordinal regression (also called ordinal classification). The problem of learning (eliciting) *real-valued* utility functions has been investigated in fields such as decision theory and economics for a long time, and has more recently become a topic of research in AI and machine learning as well. Ordinal regression has been investigated thoroughly in statistics and econometrics (McCullagh and Nelder, 1983) and has recently also received attention in machine learning. For example, a method for ordinal regression based on a modification of regression tree learning has been proposed by Kramer et al. (2001). Frank and Hall (2001) suggest a method for translating an ordinal regression problem into a set of ordinary (binary) classification problems. Herbich et al. (2000) approach ordinal regression in the context of support vector machines, using a special type of loss function suitable for comparing predictions on an ordered categorical scale. Cao-Van and De Baets (2003) consider the task to learn a *monotone* decision tree, a problem they refer to as *supervised ranking*. Here, monotonicity refers to the dependency of the (ordinal) output attribute on the input attributes, which are assumed to be *criteria* (i.e., ordinal "the higher the better" attributes).

Chajewska et al. (1998) simplify the elicitation of utility functions by clustering exemplary utility functions, deriving prototypes from the clusters, and inducing a decision tree whose inner nodes are associated with properties of utility functions (questions about a person's preferences that can be directly asked to the person) and whose leaf nodes are identified with the prototypes. The idea of Chajewska et al. (1999) is to simplify elicitation by exploiting the *additive independence* of variables. Given a database of exemplary utility functions, statistical learning (model selection) methods are used in order to induce a factorization of utility functions into additive subutility functions. Chajewska et al. (2000) accomplish learning of a utility function by treating utility as a random variable. Starting with some prior distribution (derived from analyzing a database of available utility functions), the model is incrementally updated based on information elicited from the user. In order to decide on which questions should be asked next to the user, the authors fall back on the principle underlying the *value of information*. Chajewska et al. (2001) study the problem of learning the utility function that determines the behavior of an agent which is rational in the sense of expected utility theory. The approach proposed by the authors proceeds from a prior probability distribution over a class of utility functions having a certain (linear) structure. The agent's decisions are then used for defining constraints on its true utility function (see (Ng and Russell, 2000) for a quite similar approach). Finally, these constraints are employed in order to turn the prior distribution over the class of utility functions into a posterior distribution.

Learning preferences is also a key topic in recommender systems and collaborative filtering (Goldberg et al., 1992; Resnick and Varian, 1997; Kautz, 1998). Methods proposed in this field are closer to learning utility functions, but are often specifically adjusted to commercial applications where the set of alternatives (labels) to be recommended is usually very large. The method of choice is quite often a case-based or memory-based approach, where the basic idea is to estimate a user's preferences from the preferences of other users that appear to be *similar* (see, e.g., Ha and Haddawy, 2003; Nakamura and Abe, 1998; Billsus and Pazzani, 1998).

Aiolli (2005) proposed an interesting general framework that allows to specify both qualitative and quantitative preference constraints on an underlying utility function. In addition to the pairwise preference constraints that we also use (which he interprets as constraints on a utility function), Aiolli (2005) 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.

7. Conclusions

In this paper, we have given a systematic overview of recent research activities in *preference learning*, a subfield of machine learning which is concerned with the problem of learning preference models from empirical preference data. Certainly, preference learning thus defined is of major importance for research on preferences in artificial intelligence, as it complements methods for the modeling of and the reasoning with preferences.

Focusing on a particular type of preference model, namely rankings, we have suggested a learning algorithm for a problem called *label ranking*. The merits of our method, 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. 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 4.2). Thirdly, RPC is superior to alternative approaches with regard to efficiency and computational complexity, as we have shown both theoretically and experimentally (cf. Sections 4.1 and 5).

Research in preference learning has only started, and many interesting problems remain to be solved. For example, recall our discussion about the connection between the ranking procedure used in RPC and risk minimization with regard to a particular loss function. We already know that some loss functions can be minimized by using a suitable ranking procedure, whereas the nature of the pairwise approach makes risk minimization principally impossible for other distance measures. What is missing, however, is a more complete characterization of the type of loss functions that can can be minimized and those that cannot. Another open issue is an extension of label ranking 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.

References

Fabio Aiolli A preference model for structured supervised learning tasks. In *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM-05)*, pages 557–560. IEEE Computer Society, 2005.

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

N. Alon. Ranking Tournaments. *SIAM Journal on Discrete Mathematics* 20(1), pages 137–142.

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

Michael O. Ball and Ulrich Derigs. An analysis of alternative strategies for implementing matching algorithms. *Networks*, 13:517–549, 1983.

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

Daniel Billsus and Michael Pazzani. Learning collaborative information filters. In *Proceedings of the 15th International Conference on Machine Learning (ICML-98)*, pages 46–54. Morgan Kaufmann, 1998.

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

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 21:135–191.

Ralph A. Bradley and Milton E. Terry The rank analysis of incomplete block designs — I. The method of paired comparisons. *Biometrika*, 39:324–345, 1952.

Steven J. Brams and Peter C. Fishburn. Voting procedures. In K. J. Arrow, A. K. Sen, and K. Suzumura, editors, *Handbook of Social Choice and Welfare (Vol. 1)*, chapter 4. Elsevier, 2002.

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

Kim Cao-Van and Bernard De Baets. Growing decision trees in an ordinal setting. *International Journal of Intelligent Systems* 18, pages 733-750, 2003.

Urszula Chajewska, Lise Getoor, Joseph Norman, and Yuval Shahar. Utility elicitation as a classification problem. In G. F. Cooper and S. Moral, editors, *Proceedings of the 14th Conference on Uncertainty in AI (UAI-98)*, pages 79–88, 1998.

Urszula Chajewska, M. Kuppermann, and Daphne Koller. Discovering the structure of utility functions based on additive and conditionally additive independence properties between utility attributes. In *Proceedings of the 21st Annual Meeting of the Society for Medical Decision Making (MDM-99)*, 1999.

Urszula Chajewska, Daphne Koller, and Ronald Parr. Making rational decisions using adaptive utility elicitation. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00)*, pages 363–369, 2000.

Urszula Chajewska, Daphne Koller, and Dirk Ormoneit. Learning an agent's utility function by observing behavior. In *Proceedings of the 18th International Conference on Machine Learning (ICML-01)*, pages 35–42, 2001.

William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.

D. Coppersmith, L. Fleischer, and A. Rudra. Ordering by weighted number of wins gives a good ranking for weighted tournaments. *ACM-SIAM Symposium on Discrete Algorithms* (SODA), 776–782, 2006.

Koby Crammer and Yoram Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2003a.

Koby Crammer and Yoram Singer. A family of additive online algorithms for category ranking. *Journal of Machine Learning Research*, 3:1025–1058, 2003b.

Ofer Dekel, Christopher D. Manning, and Yoram Singer. Log-Linear Models for Label Ranking. In S. Thrun, L. K. Saul, and B. Schölkopf (eds.) *Advances in Neural Information Processing Systems 16 (NIPS-2003)*, MIT Press 2004.

Jon Doyle. Prospects for preferences. *Computational Intelligence*, 20(2):111–136, 2004.

Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation methods for the Web. In *Proceedings of the 10th International World Wide Web Conference*, pages 613–622, 2001.

János Fodor and Marc Roubens. *Fuzzy Preference Modelling and Multicriteria Decision Support*. Kluwer Academic Publishers, 1994.

Eibe Frank and Mark Hall. A simple approach to ordinal classification. In L. De Raedt and P. Flach, editors, *Proceedings of the 12th European Conference on Machine Learning (ECML-01)*, pages 145–156, Freiburg, Germany, 2001. Springer-Verlag.

Jerome H. Friedman. Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University, Stanford, CA, 1996.

Johannes Fürnkranz. Round robin ensembles. *Intelligent Data Analysis*, 7(5):385–404, 2003.

Johannes Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2:721–747, 2002.

Johannes Fürnkranz and Eyke Hüllermeier. Pairwise preference learning and ranking. In N. Lavrač, D. Gamberger, H. Blockeel, and L. Todorovski, editors, *Proceedings of the 14th European Conference on Machine Learning (ECML-03)*, volume 2837 of *Lecture Notes in Artificial Intelligence*, pages 145–156, Cavtat, Croatia, 2003. Springer-Verlag.

Johannes Fürnkranz and Eyke Hüllermeier. Preference learning. *Künstliche Intelligenz*, 19 (1):60–61, 2005.

David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave and information tapestry. *Communications of the ACM*, 35(12):61–70, December 1992.

Vu Ha and Peter Haddawy. Similarity of personal preferences: Theoretical foundations and empirical analysis. *Artificial Intelligence*, 146:149–173, 2003.

Peter Haddawy, Vu Ha, Angelo Restificar, Benjamin Geisler, and John Miyamoto. Preference elicitation via theory refinement. *Journal of Machine Learning Research*, 4:317–337, 2003.

Sariel Har-Peled, Dan Roth, and Dav Zimak. Constraint classification: A new approach to multiclass classification. In N. Cesa-Bianchi, M. Numao, and R. Reischuk, editors, *Proceedings of the 13th International Conference on Algorithmic Learning Theory (ALT-02)*, pages 365–379, Lübeck, Germany, 2002. Springer.

Sariel Har-Peled, Dan Roth, and Dav Zimak. Constraint classification for multiclass classification and ranking. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15 (NIPS-02)*, pages 785–792, 2003.

Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling. In M.I. Jordan, M.J. Kearns, and S.A. Solla, editors, *Advances in Neural Information Processing Systems 10 (NIPS-97)*, pages 507–513. MIT Press, 1998.

Ralf Herbich, Thore Graepel, and Klaus Obermayer. Large margin rank boundaries for ordinal regression. In A. J. Smola, P. J. Bartless, B Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 115–132. MIT Press, 2000.

Ralf Herbich and Thore Graepel. Large scale bayes point machines. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13 (NIPS 2000)*, pages 528–534. MIT Press, 2001.

Ralf Herbrich, Thore Graepel, Peter Bollmann-Sdorra, and Klaus Obermayer. Supervised learning of preference relations. In *Proceedings des Fachgruppentreffens Maschinelles Lernen (FGML-98)*, pages 43–47, 1998.

Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, March 2002.

Eyke Hüllermeier and 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, 2004a.

Eyke Hüllermeier and Johannes Fürnkranz. Learning label preferences: Ranking error versus position error. In *Advances in Intelligent Data Analysis: Proceedings of the 6th International Symposium (IDA-05)*, pages 180–191. Springer-Verlag, 2005.

Eyke Hüllermeier and 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, 2004b.

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)*, pages 133–142. ACM Press, 2002.

Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and 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.

Henry Kautz, editor. *Recommender Systems: Papers from the AAAI Workshop*, Menlo Park, CA, 1998. AAAI Press. Technical Report WS-98-08.

Maurice G. Kendall. *Rank correlation methods*. Charles Griffin, London, 1955.

Erich-Peter Klement, Radko Mesiar and Endre Pap. *Triangular Norms*. Kluwer Academic Publishers, 2002.

Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. Single-layer learning revisited: A stepwise procedure for building and training a neural network. In F. Fogelman Soulié and J. Hérault, editors, *Neurocomputing: Algorithms, Architectures and Applications*, volume F68 of *NATO ASI Series*, pages 41–50. Springer-Verlag, 1990.

Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. Handwritten digit recognition by neural networks with single-layer training. *IEEE Transactions on Neural Networks*, 3(6): 962–968, 1992.

Stefan Kramer, Gerhard Widmer, Bernhard Pfahringer, and Michael DeGroeve. Prediction of ordinal classes using regression trees. *Fundamenta Informaticae*, XXI:1001–1013, 2001.

Ulrich H.-G. Kreßel. Pairwise classification and support vector machines. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, chapter 15, pages 255–268. MIT Press, Cambridge, MA, 1999.

Erich L. Lehmann and H. J. M. D'Abrera. *Nonparametrics: Statistical Methods Based on Ranks, rev. ed.* Prentice-Hall, Englewood Cliffs, NJ, 1998.

Bao-Liang Lu and 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):1244–1256, September 1999.

Peter McCullagh and John A. Nelder. *Generalized Linear Models.* Chapman & Hall, London, 1983.

Donald Michie, David J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, 1994. Data available at `ftp.ncc.up.pt/pub/statlog/`.

H. Naessens, H. De Meyer, and B. De Baets. Algorithms for the computation of T-transitive closures. *IEEE Trans. on Fuzzy Systems* 10:541–551, 2002.

Atsuyoshi Nakamura and Naoki Abe. Collaborative filtering using weighted majority prediction algorithms. In *Proceedings of the 15th International Conference on Machine Learning (ICML-98)*, pages 395–403. Morgan Kaufmann, 1998.

Andrew Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning (ICML-00)*, 2000.

John Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In A.J. Smola, P. Bartlett, B. Schoelkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74, Cambridge, MA, 1999. MIT Press.

David Price, Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. Pairwise neural network classifiers with probabilistic outputs. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7 (NIPS-94)*, pages 1109–1116. MIT Press, 1995.

Dorian Pyle. *Data Preparation for Data Mining.* Morgan Kaufmann, San Francisco, CA, 1999.

Filip Radlinski and Thorsten Joachims. Learning to rank from implicit feedback. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD-05)*, 2005.

Paul Resnick and Hal R. Varian. Special issue on recommender systems. *Communications of the ACM*, 40(3), 1997.

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

Michael S. Schmidt and Herbert Gish. Speaker identification via support vector classifiers. In *Proceedings of the 21st IEEE International Conference Conference on Acoustics, Speech, and Signal Processing (ICASSP-96)*, pages 105–108, Atlanta, GA, 1996.

B. Schweizer and A. Sklar. *Probabilistic Metric Spaces*, North-Holland, New York, 1983.

Charles Spearman. The proof and measurement of association between two things. *American Journal of Psychology*, 15:72–101, 1904.

Gerald Tesauro. Connectionist learning of expert preferences by comparison training. In D. Touretzky, editor, *Advances in Neural Information Processing Systems 1 (NIPS-88)*, pages 99–106. Morgan Kaufmann, 1989.

Nicolas Usunier, Massih-Reza Amini, and Patrick Gallinari. Generalization error bounds for classifiers trained with interdependent data. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18 (NIPS 2005)*, pages 1369–1376. MIT Press, 2006.

Jun Wang. Artificial neural networks versus natural neural networks: A connectionist paradigm for preference assessment. *Decision Support Systems*, 11:415–429, 1994.

Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools with Java implementations.* Morgan Kaufmann, San Francisco, 2000.

Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5(Aug):975–1005, 2004.

Appendix A.

approach	1	2	3	4	5
RPC (bin-vote)	0.116 ±0.009	0.112 ±0.009	0.120 ±0.012	0.188 ±0.011	0.285 ±0.008
RPC (soft-vote)	**0.122** ±0.009	**0.114** ±0.009	0.145 ±0.011	**0.196** ±0.010	**0.327** ±0.007
CC	0.114 ±0.008	0.107 ±0.009	**0.148** ±0.011	0.186 ±0.010	0.287 ±0.008

Table 4: Experimental results: Real-world Setting (Spearman rank correlation)

approach	1	2	3	4	5
RPC (bin-vote)	0.146 ±0.011	0.134 ±0.011	0.176 ±0.013	0.218 ±0.012	0.356 ±0.010
RPC (soft-vote)	**0.158** ±0.011	**0.141** ±0.011	**0.179** ±0.013	**0.242** ±0.012	**0.414** ±0.009
CC	0.151 ±0.011	0.126 ±0.011	0.172 ±0.013	0.218 ±0.012	0.364 ±0.010

Table 5: Experimental results: Real-world Setting (Kendall tau)

approacy8h	iris	wine	glass	vowel	vehicle
RPC (bin-vote)	0.850 ±0.019	**0.952** ±0.013	**0.891** ±0.017	0.730 ±0.013	0.861 ±0.005
RPC (soft-vote)	**0.890** ±0.019	**0.952** ±0.017	0.878 ±0.017	**0.732** ±0.011	**0.865** ±0.005
CC	0.847 ±0.017	0.944 ±0.014	0.874 ±0.016	0.723 ±0.012	0.863 ±0.003

Table 6: Experimental results (Spearman rank correlation)

approach	iris	wine	glass	vowel	vehicle
RPC (bin-vote)	0.817 ±0.026	**0.940** ±0.018	**0.882** ±0.012	0.707 ±0.012	0.835 ±0.008
RPC (soft-vote)	0.833 ±0.024	0.918 ±0.015	0.846 ±0.019	**0.718** ±0.013	0.828 ±0.008
CC	**0.847** ±0.019	0.938 ±0.015	0.865 ±0.016	0.708 ±0.012	**0.839** ±0.006

Table 7: Experimental results for 50% missing labels (Spearman rank correlation)

approach	iris	wine	glass	vowel	vehicle
RPC (bin-vote)	0.780 ±0.036	**0.907** ±0.018	0.794 ±0.019	0.683 ±0.011	**0.816** ±0.010
RPC (soft-vote)	0.760 ±0.032	0.856 ±0.024	0.739 ±0.021	**0.694** ±0.012	0.799 ±0.014
CC	**0.800** ±0.035	0.829 ±0.016	**0.817** ±0.014	0.691 ±0.010	0.815 ±0.014

Table 8: Experimental results for 70% missing labels (Spearman rank correlation)

approach	iris	wine	glass	vowel	vehicle
RPC (bin-vote)	0.791 ±0.028	0.933 ±0.021	**0.866** ±0.016	**0.637** ±0.012	**0.825** ±0.006
RPC (soft-vote)	**0.853** ±0.016	**0.936** ±0.018	0.851 ±0.015	0.619 ±0.010	0.823 ±0.006
CC	0.796 ±0.027	**0.936** ±0.017	0.829 ±0.018	0.596 ±0.011	0.810 ±0.006

Table 9: Experimental results (Kendall tau)

approach	iris	wine	glass	vowel	vehicle
RPC (bin-vote)	0.747 ±0.023	**0.936** ±0.020	**0.846** ±0.014	**0.603** ±0.010	0.790 ±0.008
RPC (soft-vote)	0.787 ±0.024	0.910 ±0.024	0.818 ±0.017	0.586 ±0.011	0.793 ±0.007
CC	**0.796** ±0.026	0.914 ±0.017	0.824 ±0.016	0.575 ±0.010	**0.795** ±0.006

Table 10: Experimental results for 50% missing labels (Kendall tau)

approach	iris	wine	glass	vowel	vehicle
RPC (bin-vote)	0.724 ±0.037	**0.888** ±0.016	0.763 ±0.024	**0.575** ±0.010	0.770 ±0.01
RPC (soft-vote)	0.716 ±0.040	0.849 ±0.016	0.685 ±0.022	0.554 ±0.011	0.746 ±0.01
CC	**0.764** ±0.025	0.798 ±0.015	**0.771** ±0.014	0.562 ±0.009	**0.771** ±0.015

Table 11: Experimental results for 70% missing labels (Kendall tau)