Labelwise versus Pairwise Decomposition in Label Ranking

Weiwei Cheng, Sascha Henzgen and Eyke Hüllermeier
Computational Intelligence Group, University of Marburg
Marburg, Germany
{cheng,henzgen,eyke}@mathematik.uni-marburg.de

Abstract

Label ranking is a specific type of preference learning problem, namely the problem of learning a model that maps instances to rankings over a finite set of predefined alternatives (labels). State-of-the-art approaches to label ranking include decomposition techniques that reduce the original problem to binary classification; ranking by pairwise comparison (RPC), for example, constructs one binary problem for each pair of alternatives. In general, each classification example refers to the *pairwise* comparison of two alternatives in a ranking. In this paper, we introduce a new (meta) learning technique for label ranking, which is based on a *labelwise* instead of a pairwise decomposition. The basic idea is to train one model per class label, namely a model that maps instances to ranks. Instead of a quadratic number of binary problems, like in RPC, this obviously gives rise to a linear number of ordinal classification problems. We propose a generalization of this approach for the practically relevant case in which the training data only contains incomplete rankings, that is, rankings of some but not all alternatives; in this case, only imprecise information about the rank of individual labels can be derived. Moreover, we provide an experimental study, in which the pairwise and the labelwise decomposition techniques are compared in a systematic way.

1 Introduction

Preference learning is an emerging subfield of machine learning, which deals with the induction of preference models from observed or revealed preference information [10]. Such models are typically used for prediction purposes, for example, to predict context-dependent preferences of individuals on various choice alternatives. Depending on the representation of preferences, individuals, alternatives, and contexts, a large variety of preference models are conceivable, and many such models have already been studied in the literature.

A specific type of preference learning problem is the problem of *label ranking*, namely the problem of learning a model that maps instances to rankings (total orders) over a finite set of predefined alternatives (labels). Several methods for label ranking have already been proposed in the literature [18]. Most of these methods are *reduction techniques* transforming the original learning task into one or several binary classification tasks. Moreover, all existing methods are *relational* in so far as they seek to learn from *relative* or *comparative* preferences, such as pairwise comparisons between alternatives [15]. Since a ranking of alternatives, by its very nature, does indeed inform about *relative* and not about *absolute* preferences, the prevalence of the relational approach is of course completely understandable.

On the other hand, since the number of alternatives in a label ranking problem is fixed, a ranking is uniquely defined by the position (rank) of each of the alternatives, which can be seen as *absolute* preference information. Admittedly, as will be explained in more detail later on, this positional information is not always readily available for training. Yet, it is arguably a bit surprising that, to the best of our knowledge, an approach focused on the learning and prediction of absolute preferences has not even been tried so far.

In this paper, we introduce an approach of that kind, namely a new meta-learning technique for label ranking, which is based on a *labelwise* instead of a *pairwise* decomposition. The basic idea is to train one model per class label, namely a model that maps instances to ranks. In other words, given a new query instance, the idea is to predict the rank of each individual label right away. Unlike existing decomposition techniques, in which the reducts are binary classification problems, this approach leads to a linear number of ordered multi-class problems.

The paper is organized as follows. The next section provides some background of the label ranking problem, and Section 3 reviews existing methods for tackling this problem. Our new approach based on labelwise decomposition (LWD) is introduced in Section 4. Section 5 is devoted to a general discussion of similarities and differences between reduction techniques for label ranking. In Section 6, we provide an experimental study, in which LWD is compared with existing decomposition techniques in a systematic way. The paper ends with some concluding remarks in Section 7.

2 Label Ranking

Let $\mathcal{Y} = \{y_1, \ldots, y_K\}$ be a finite set of (choice) alternatives; adhering to the terminology commonly used in supervised machine learning, and accounting for the fact that label ranking can be seen as an extension of multi-class classification, the y_i are also called *class labels*. We consider total order relations $\succ$ on $\mathcal{Y}$, that is, complete, transitive, and antisymmetric relations, where $y_i \succ y_j$ indicates that y_i precedes y_j in the order. Since a ranking can be seen as a special type of preference relation, we shall also say that $y_i \succ y_j$ indicates a preference for y_i over y_j.

Formally, a total order $\succ$ can be identified with a permutation $\bar{\pi}$ of the set $[K] = \{1, \ldots, K\}$, such that $\bar{\pi}(i)$ is the position of y_i in the order. We denote the class of permutations of $[K]$ (the symmetric group of order K) by $\mathbb{S}_K$. By abuse of terminology, though justified in light of the above one-to-one correspondence, we refer to elements $\bar{\pi} \in \mathbb{S}_K$ as both permutations and rankings.

In the setting of label ranking, preferences on $\mathcal{Y}$ are "contextualized" by instances $\boldsymbol{x} \in \mathbb{X}$, where $\mathbb{X}$ is an underlying instance space. Thus, each instance $\boldsymbol{x}$ is associated with a ranking $\succ_{\boldsymbol{x}}$ of the label set $\mathcal{Y}$ or, equivalently, a permutation $\bar{\pi}_{\boldsymbol{x}} \in \mathbb{S}_K$. More specifically, since label rankings do not necessarily depend on instances in a deterministic way, each instance $\boldsymbol{x}$ is associated with a probability distribution $\mathbf{P}(\cdot \,|\, \boldsymbol{x})$ on $\mathbb{S}_K$. Thus, for each $\bar{\pi} \in \mathbb{S}_K$, $\mathbf{P}(\bar{\pi} \,|\, \boldsymbol{x})$ denotes the probability to observe the ranking $\bar{\pi}$ in the context specified by $\boldsymbol{x}$.

As an illustration, suppose $\mathbb{X}$ is the set of people characterized by attributes such as sex, age, profession, and marital status, and labels are music genres: $\mathcal{Y} = \{\texttt{Rock}, \texttt{Pop}, \texttt{Classic}, \texttt{Jazz}\}$. Then, for $\boldsymbol{x} = (m, 30, \text{teacher}, \text{married})$ and $\bar{\pi} = (2, 1, 4, 3)$, $\mathbf{P}(\bar{\pi} \,|\, \boldsymbol{x})$ denotes the probability that a 30 years old married man, who is a teacher, prefers Pop music to Rock to Classic to Jazz.

2.1 The Label Ranking Problem

The goal in label ranking is to learn a "label ranker", that is, a model

$$\mathcal{M} : \mathbb{X} \longrightarrow \mathbb{S}_K$$

that predicts a ranking $\hat{\pi}$ for each instance $\boldsymbol{x}$ given as an input. More specifically, seeking a model with optimal prediction performance, the goal is to find a risk (expected loss) minimizer

$$\mathcal{M}^* \in \operatorname*{argmin}_{\mathcal{M} \in \mathbf{M}} \int_{\mathbb{X} \times \mathbb{S}_K} D(\mathcal{M}(\boldsymbol{x}), \bar{\pi}) \, d\mathbf{P} \enspace ,$$

where $\mathbf{M}$ is the underlying model class, $\mathbf{P}$ is the joint measure $\mathbf{P}(\boldsymbol{x}, \bar{\pi}) = \mathbf{P}(\boldsymbol{x})\mathbf{P}(\bar{\pi} \,|\, \boldsymbol{x})$ on $\mathbb{X} \times \mathbb{S}_K$ and D is a loss function on $\mathbb{S}_K$; common choices of D will be introduced below.

As training data $\mathbb{D}$, a label ranker uses a set of instances $\boldsymbol{x}_n$ ($n \in [N]$), together with information about the associated rankings π_n. Ideally, complete rankings are given as training information, i.e., a single observation is a tuple of the form $(\boldsymbol{x}_n, \pi_n) \in \mathbb{X} \times \mathbb{S}_K$; we call an observation of that kind a *complete* example. From a practical point of view, however, it is important to allow for incomplete information in the form of a ranking of some but not all of the labels in $\mathcal{Y}$:

$$y_{\tau(1)} \succ_{\boldsymbol{x}} y_{\tau(2)} \succ_{\boldsymbol{x}} \cdots \succ_{\boldsymbol{x}} y_{\tau(J)} \enspace , \tag{1}$$

where $J < K$ and $\{\tau(1), \ldots, \tau(J)\} \subset [K]$. For example, for an instance $\boldsymbol{x}$, it might be known that $y_2 \succ_{\boldsymbol{x}} y_1 \succ_{\boldsymbol{x}} y_5$, while no preference information is given about the labels y_3 or y_4.

In the following, we will write complete rankings $\bar{\pi}$ with an upper bar (as we already did above). If a ranking π is not complete, then $\pi(j)$ is the position of y_j in the incomplete ranking, provided this label is contained, and $\pi(j) = 0$ otherwise; thus, if $\bar{\pi}$ is a "completion" of π, then $\bar{\pi}(k) \geq \pi(k)$ for all $k \in [K]$. In the above example (1), $\pi = (2, 1, 0, 0, 3)$. We denote by $|\pi| = \{j \,|\, \pi(j) > 0\}$ the size of the ranking; thus, π is complete if $|\pi| = K$.

2.2 Prediction Accuracy

The prediction accuracy of a label ranker is assessed by comparing the true ranking $\bar{\pi}$ with the prediction $\hat{\pi}$, using a distance measure D on rankings. Among the most commonly used measures is the Kendall distance, which is defined by the number of inversions, that is, index pairs $\{i, j\} \subset [K]$ such that the order of y_i and y_j in $\bar{\pi}$ is inverted in $\hat{\pi}$:

$$D(\bar{\pi}, \hat{\pi}) = \sum_{1 \leq i < j \leq K} [\![(\bar{\pi}(i) - \bar{\pi}(j))(\hat{\pi}(i) - \hat{\pi}(j)) < 0]\!] \tag{2}$$

The well-known Kendall rank correlation measure is an affine transformation of (2) to the range $[-1, +1]$. Besides, the sum of L_1 or L_2 losses on the ranks of the individual labels are often used as an alternative distance measures:

$$D_1(\bar{\pi}, \hat{\pi}) = \sum_{i=1}^{M} |\bar{\pi}(i) - \hat{\pi}(i)| \tag{3}$$

$$D_2(\bar{\pi}, \hat{\pi}) = \sum_{i=1}^{M} (\bar{\pi}(i) - \hat{\pi}(i))^2 \tag{4}$$

These measures are closely connected with two other well-known rank correlation measures: Spearman's footrule is an affine transformation of (3) to the interval $[-1, +1]$, and Spearman's rank correlation (Spearman's rho) is such a transformation of (4).

3 Label Ranking Methods

The arguably most straightforward way to addressing the label ranking problem is to treat it as a classification problem with $K!$ classes, considering each ranking $\bar{\pi} \in \mathbb{S}_K$ as a separate (meta-)class; this is to some extent comparable to the *label powerset* approach to multilabel classification [17], which considers each subset Y of the original label set $\mathcal{Y}$ as a new meta-class. Obviously, however, this approach comes with a number of disadvantages, making it likely to fail in practice. First of all, the number of meta-classes is even larger than for multilabel classification. For example, with only $K = 6$ labels, the resulting classification problem would consist of 720 meta-classes—there is no classifier that can handle such a number of classes in a reasonable way. Second, it is not clear how to apply this approach in the case of incomplete observations (1). Third, by treating each meta-class as a separate category, this approach fails to exploit the structure on the output space $\mathbb{S}_K$, which is induced by the underlying distance measure D.

Indeed, label ranking can be seen as a specific type of *structured output* prediction [1], namely the problem to predict structures in the form of permutations. In the literature, several methods for label ranking have been proposed that try to exploit the structure on $\mathbb{S}_K$ in one way or the other, including generalizations of standard machine learning methods such as nearest neighbor estimation [3] and decision tree learning [6], as well as statistical inference based on parametrized models of rank data [5].

Here, we are specifically interested in *reduction techniques*, that is, meta-learning techniques that reduce the original label ranking problem into one or several classification problems that are easier to solve. Among the techniques proposed so far, there are two approaches that both reduce label ranking to binary classification, albeit in a different way. Whereas the first technique, *constraint classification* (CC), produces a single "large" classification problem, the second one, *ranking by pairwise comparison* (RPC), yields a quadratic (in K) number of "small" binary problems. In the following, both approaches will be presented in more detail.

3.1 Constraint Classification

Constraint classification [12] is based on the idea of learning value functions $f_k : \mathbb{X} \longrightarrow \mathbb{R}$, one for each label y_k ($k \in [K]$), that estimate a (latent) degree of utility of y_k in the context specified by an instance. Given such functions, a prediction $\hat{\pi}$ for a new query instance $\boldsymbol{x}$ is then simply obtained by sorting the labels in decreasing order of their (estimated) utility:

$$\hat{\pi} = \operatorname*{argsort}_{k \in [K]} f_k(\boldsymbol{x}) \tag{5}$$

More specifically, assuming $\mathbb{X} = \mathbb{R}^d$, the value functions are taken as linear functions of the form

$$f_k(\boldsymbol{x}) = f_k(x_1, \ldots, x_d) = \sum_{i=1}^{d} \alpha_{k,i} x_i \tag{6}$$

with label-specific coefficients $\alpha_{k,i}$ ($i \in [d]$).

Now, a pairwise preference $y_k \succ_{\boldsymbol{x}} y_j$ between two labels translates into the constraint $f_k(\boldsymbol{x}) - f_j(\boldsymbol{x}) > 0$ or, equivalently, $f_j(\boldsymbol{x}) - f_k(\boldsymbol{x}) < 0$. Both constraints, the positive and the negative one, can be expressed in terms of the sign of an inner product $\langle \boldsymbol{z}, \boldsymbol{\alpha} \rangle$, where

$$\boldsymbol{\alpha} = (\alpha_{1,1}, \ldots, \alpha_{1,d}, \alpha_{2,1}, \ldots, \alpha_{2,d}, \ldots, \alpha_{K,1}, \ldots, \alpha_{K,d})$$

is a concatenation of all label-specific coefficients. Correspondingly, the vector $\boldsymbol{z}$ is constructed by mapping the original d-dimensional training example $\boldsymbol{x} = (x_1, \ldots, x_d)$ into an $(K \times d)$-dimensional space: For the positive constraint, $\boldsymbol{x}$ is copied into the components $((k-1) \times d + 1), \ldots, (k \times d)$ and its negation $-\boldsymbol{x}$ into the components $((j-1) \times d+1), \ldots, (j \times d)$; the remaining entries are filled with 0. For the negative constraint, a vector is constructed with the same elements but reversed signs. Both constraints can be considered as training examples for a conventional binary classifier in a $(K \times d)$-dimensional space: The first vector is a positive and the second one a negative example.

CC constructs training examples of that kind by splitting observed rankings into pairwise preferences. More specifically, an incomplete ranking (1) is split into $J-1$ preferences $y_{\tau(k)} \succ_{\boldsymbol{x}} y_{\tau(k+1)}$ ($k \in [J-1]$), and each of these preferences is turned into a positive and a negative example for the binary classifier as described above. The corresponding binary classification problem can then be tackled by standard methods for fitting a separating hyperplane in this space, that is, a suitable vector $\boldsymbol{\alpha}$ satisfying as many as possible constraints.

3.2 Ranking by Pairwise Comparison

Ranking by pairwise comparison [15] is an extension of pairwise classification [9], an established technique for reducing multi-class to binary classification. In the setting of label ranking, RPC trains one model $\mathcal{M}_{i,j} : \mathbb{X} \longrightarrow [0,1]$ for each pair of labels $\{y_i, y_j\}$; thus, $K(K-1)/2$ such models are needed in total. Given instance $\boldsymbol{x}$ as input, the model $\mathcal{M}_{i,j}$ is supposed to predict the probability of $y_i \succ_{\boldsymbol{x}} y_j$, i.e., $\mathcal{M}_{i,j}(\boldsymbol{x})$ is an estimation of the probability $\mathbf{P}(\pi(i) < \pi(j) \,|\, \boldsymbol{x})$.

The data $\mathbb{D}_{i,j}$ used to train $\mathcal{M}_{i,j}$ is constructed from the original data $\mathbb{D}$ as follows: If $\boldsymbol{x}_n$ is an instance in $\mathbb{D}$ that has been observed together with a possibly incomplete ranking of labels in $\mathcal{Y}$, then

- $\boldsymbol{x}_n$ is added as a positive example to $\mathbb{D}_{i,j}$ if the ranking contains both y_i and y_j, and the former precedes the latter;
- $\boldsymbol{x}_n$ is added as a negative example to $\mathbb{D}_{i,j}$ if the ranking contains both y_i and y_j, and the latter precedes the former;
- $\boldsymbol{x}_n$ is ignored if either y_i or y_j (or both) are missing in the ranking.

Once $\mathbb{D}_{i,j}$ has been constructed, any method for (probabilistic) binary classification can be used to induce the model $\mathcal{M}_{i,j}$.

At prediction time, when a ranking $\hat{\pi}$ needs to be predicted for a new instance $\boldsymbol{x}$, this instance is first submitted to each of the models $\mathcal{M}_{i,j}$ ($1 \le i < j \le K$), and the predictions of these models are combined into a (weighted) preference relation

$$P = \begin{bmatrix} - & p_{1,2} & p_{1,3} & \cdots & p_{1,K} \\ p_{2,1} & - & p_{2,3} & \cdots & p_{2,K} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{K,1} & p_{K,2} & p_{K,3} & \cdots & - \end{bmatrix} , \tag{7}$$

where

$$p_{i,j} = \begin{cases} \mathcal{M}_{i,j}(\boldsymbol{x}) & \text{if } i < j \\ 1 - \mathcal{M}_{j,i}(\boldsymbol{x}) & \text{if } j < i \end{cases} .$$

The preference relation (7) does normally not suggest a ranking $\hat{\pi}$ in an unequivocal way: Since the binary models $\mathcal{M}_{i,j}$ are trained independently of each other, and the predictions $p_{i,j}$ are not necessarily perfect, P may exhibit inconsistencies such as preferential cycles. What is needed, in general, is a *ranking procedure* that turns P into a ranking $\hat{\pi}$.

The standard approach in RPC is to apply a weighted voting procedure, in which the labels are sorted according to the sum of weighted votes in their own favor:

$$\hat{\pi} = \operatorname*{argsort}_{k \in [K]} s_k(\boldsymbol{x}), \tag{8}$$

where

$$s_k(\boldsymbol{x}) = \sum_{1 \le j \ne k \le K} p_{k,j} .$$

Under certain technical assumptions (the $p_{i,j}$ are independent and unbiased estimations of $\mathbf{P}(\pi(i) < \pi(j) \,|\, \boldsymbol{x})$), it can be shown that the prediction (8) is minimizing the expected loss with respect to (4). For other loss functions, other ranking procedures might be optimal.

4 Labelwise Decomposition

In this section, we introduce a new meta-learning technique for label ranking, which is based on the idea of reducing the original problem to standard classification problems in a *labelwise* manner.

4.1 The Case of Complete Training Information

If the training data $\mathbb{D}$ consists of complete examples $(\boldsymbol{x}_n, \bar{\pi}_n)$, then each such example informs about the rank $\bar{\pi}(k)$ of the label y_k in the ranking associated with $\boldsymbol{x}_n$. Thus, a quite natural idea is to learn a model

$$\mathcal{M}_k : \mathbb{X} \longrightarrow [K]$$

that predicts the rank of y_k, given an instance $\boldsymbol{x} \in \mathbb{X}$ as an input. Indeed, such a model can be trained easily on the data

$$\mathbb{D}_k = \Big\{ (\boldsymbol{x}_n, r_n) \,|\, (\boldsymbol{x}_n, \bar{\pi}_n) \in \mathbb{D},\, r_n = \bar{\pi}_n(k) \Big\} \subset \mathbb{X} \times [K] \tag{9}$$

It is important to note, however, that the classification problem thus produced is not a binary one, like in CC and RPC. Instead, we need to solve a multi-class problem with K classes, where each class corresponds to a possible rank. More specifically, since these ranks have a natural order, we are facing an *ordinal classification* problem.

Like in RPC, we assume that a probabilistic approach is used to train the models $\mathcal{M}_k$ ($k \in [K]$). For example, if the (ordinal) classifiers are specified by a parameter $\theta \in \Theta$, $\mathcal{M}_k$ could be identified by the maximum likelihood estimate

$$\theta_k = \operatorname*{argmax}_{\theta \in \Theta} \prod_{n=1}^{N} \mathbf{P}(r_n \,|\, \boldsymbol{x}_n, \theta) \ . \tag{10}$$

Then, given a new query instance $\boldsymbol{x}$, each of these models is supposed to predict a probability distribution

$$\mathcal{M}_k(\boldsymbol{x}) = \big(p_{k,1}, p_{k,2}, \ldots, p_{k,K}\big) \in [0,1]^K \ , \tag{11}$$

where $p_{k,j} = \mathbf{P}(\bar{\pi}(k) = j \,|\, \boldsymbol{x})$ is the (predicted) probability that y_k is on rank j.

4.2 Aggregation

As we have seen in previous sections, each reduction techniques also involves an *aggregation procedure*, which is responsible for combining the predictions of the classification models into a ranking $\hat{\pi}$. In the case of CC and RPC, these aggregations are given by the sorting procedures (5) and (8), respectively.

Consider a loss function D on $\mathbb{S}_K$ that is labelwise decomposable, i.e., which can be written in the form

$$D(\bar{\pi}, \hat{\pi}) = \sum_{k=1}^{K} D_k(\bar{\pi}(k), \hat{\pi}(k)).$$

Obviously, the L_1 and L_2 loss (3) and (4) are both of this type. Then, given probabilities of the form (11), the expected loss caused by a prediction $\hat{\pi}$ can be written as

$$\begin{aligned}\mathbb{E}\big(D(\bar{\pi}, \hat{\pi})\big) &= \sum_{k=1}^{K} \mathbb{E}\big(D_k(\bar{\pi}(k), \hat{\pi}(k))\big) \qquad (12)\\ &= \sum_{k=1}^{K}\sum_{j=1}^{K} D_k(j, \hat{\pi}(k)) \cdot p_{k,j} \\ &= \sum_{k=1}^{K} L_k(\hat{\pi}(k)) \ ,\end{aligned}$$

where $L_k(r)$ is the cost of putting y_k on position r, namely the loss expected on y_k when assigning this label to position r in the ranking $\hat{\pi}$. In the case of (3), for example, this cost is given by

$$L_k(r) = \sum_{j=1}^{K} |j - r| \cdot p_{k,j} \ .$$

Thus, an optimal solution would consists of assigning y_k the position $\hat{\pi}(k) = r$ for which $L_k(r)$ is minimal. However, noting that each position $r \in [K]$ must be assigned at most once, this approach is obviously not guaranteed to produce a feasible solution. Instead, the minimization of (12) requires the solution of an *optimal assignment problem* [4]:

- labels $y_k \in \mathcal{Y}$ must be uniquely assigned to ranks $r = \hat{\pi}(k) \in [K]$;
- assigning y_k to rank r causes a cost of $L_k(r)$;
- the goal is to minimize the sum of all assignment costs.

Assignment problems of that kind have been studied extensively in the literature, and efficient algorithms for their solution are available. The well-known Hungarian algorithm [16], for example, solves the above problem in time $O(K^3)$. Such algorithms can be used to produce a (risk minimizing) prediction $\hat{\pi}$ on the basis of probabilistic predictions (11).

4.3 The Case of Incomplete Training Information

As mentioned before, the original training data $\mathbb{D}$ is not necessarily supposed to contain complete rank information; instead, for a training instance $\boldsymbol{x}_n$, only an incomplete ranking π_n of a subset of the labels in $\mathcal{Y}$ might have been observed, while the complete ranking $\bar{\pi}_n$ is not given. In this case, the above method is not directly applicable: If at least one label is missing, i.e., $|\pi_n| < K$, then none of the true ranks $\bar{\pi}_n(k)$ is precisely known; consequently, the training data (9) cannot be constructed.

Nevertheless, even in the case of incomplete rankings, non-trivial information can be derived about the rank $\bar{\pi}(k)$ for at least some of the labels y_k. In fact, if $|\pi| = J$ and $\pi(k) = r > 0$, then

$$\bar{\pi}(k) \in \big\{r, r+1, \ldots, r+K-J\big\} \ .$$

Of course, if $\pi(k) = 0$ (i.e., y_k is not present in the ranking), only the trivial information $\bar{\pi}(k) \in [K]$ can be derived. Yet, more precise information can be obtained under additional assumptions. For example, if π is known to be the top of the ranking $\bar{\pi}$, then

$$\begin{cases} \bar{\pi}(k) = \pi(k) & \text{if } \pi(k) > 0 \\ \bar{\pi}(k) \in \{J+1, \ldots, K\} & \text{if } \pi(k) = 0 \end{cases} . \tag{13}$$

This scenario is highly relevant, since top-ranks are observed in many practical applications.

In general, the type of training data that can be derived for a label y_k in the case of incomplete rank information are examples of the form

$$\big(\boldsymbol{x}_n, R_n\big) \in \mathbb{X} \times 2^{[K]} \ , \tag{14}$$

that is, an instance $\boldsymbol{x}_n$ together with a set of possible ranks R_n. The problem of learning from data with *imprecise* class information has recently been studied in the literature, where it is called learning from *ambiguously labeled examples* [13] or learning from *partial labels* [11; 7]. As explained in [13], a reasonable approach to learning from imprecise data is to combine model identification and *data disambiguation*, that is, trying to fit an optimal model while simultaneously finding the "true data". Again adopting the principle of maximum likelihood inference, one way to realize this idea is to maximize a generalized likelihood function:

$$\theta_k = \operatorname*{argmax}_{\mathbf{r} \in \mathcal{R},\, \theta \in \Theta} \prod_{n=1}^{N} \mathbf{P}(r_n \,|\, \boldsymbol{x}_n, \theta) \ , \tag{15}$$

where $\mathcal{R}$ is the set of all selections of the rank information (14), that is, the set of all vectors $\mathbf{r} = (r_1, \ldots, r_N) \in [K]^N$ such that $r_n \in R_n$.

4.4 Probabilistic Modeling of Missing Label Information

Under additional assumptions about the process that eliminates labels from a complete ranking $\bar{\pi}$, this approach can be further refined. For example, under the "missing at random" assumption, according to which the $K - J$ labels that are missing have been selected uniformly at random from the set of all K labels, the probability to observe $\pi(k) = j > 0$ is given by

$$\sum_{r=j}^{j+K-J} \mathbf{P}(r \,|\, \boldsymbol{x}, \theta_k) \frac{\binom{r-1}{r-j}\binom{K-r}{K-J-r+j}}{\binom{K}{J}} \ . \tag{16}$$

Each term in (16) expresses the probability that the true rank of y_k in $\bar{\pi}$ is r, $r - j$ labels are removed "above" y_k (thus bringing it to position j), and the other $K - J - r + j$ labels are removed "below" y_k. The probability to observe $\pi(k) = 0$ is given by

$$\sum_{r=1}^{K} \mathbf{P}(r \,|\, \boldsymbol{x}, \theta_k) \frac{J}{K} = \frac{J}{K} \ ,$$

i.e., by a constant that can be ignored in likelihood maximization. Thus, estimation of θ_k can be accomplished as follows:

$$\hat{\theta}_k = \underset{\theta \in \Theta}{\operatorname{argmax}} \prod_{n \in [N], \pi_n(k) > 0} \ \sum_{r=\pi_n(k)}^{\pi_n(k)+K-|\pi_n|} \mathbf{P}_{r,\pi_n(k)} \tag{17}$$

with

$$\mathbf{P}_{r,\pi_n(k)} = \mathbf{P}(r \,|\, \boldsymbol{x}, \theta) \frac{\binom{r-1}{r-\pi_n(k)}\binom{K-r}{K-|\pi_n|-r+\pi_n(k)}}{\binom{K}{|\pi_n|}}$$

Under the top-rank model (13), the probability to observe $\pi(k) = j$ is given by

$$\begin{cases} \mathbf{P}(j \,|\, \boldsymbol{x}, \theta_k) & \text{if } j > 0 \\ \sum_{r=|\pi_n|+1}^{K} \mathbf{P}(r \,|\, \boldsymbol{x}, \theta_k) & \text{if } j = 0 \end{cases} \ , \tag{18}$$

and θ_k can be estimated as follows:

$$\hat{\theta}_k = \underset{\theta \in \Theta}{\operatorname{argmax}} \prod_{n \in [N], \pi_n(k) > 0} \mathbf{P}(\pi_n(k) \,|\, \boldsymbol{x}, \theta) \tag{19}$$
$$\times \prod_{n \in [N], \pi_n(k) = 0} \ \sum_{r=|\pi_n|+1}^{K} \mathbf{P}(r \,|\, \boldsymbol{x}, \theta)$$

For our experiments in Section 6, we implemented (17) and (19) using a corresponding extension of ordinal logistic regression. Thus, the probabilities $\mathbf{P}(r \,|\, \boldsymbol{x}, \theta)$ are expressed in terms of log-linear functions. More specifically, ordinal logistic regression models ratios of the *cumulative distribution*:

$$\log\left(\frac{c_k(\boldsymbol{x})}{1 - c_k(\boldsymbol{x})}\right) = \beta_k + \boldsymbol{w}^\top \boldsymbol{x} \tag{20}$$

for $k \in [K-1]$, where $c_k(\boldsymbol{x}) = \mathbf{P}(r \le k \,|\, \boldsymbol{x})$ is the (conditional) probability of a rank $\le k$ (hence $\mathbf{P}(r \,|\, \boldsymbol{x}, \theta) = c_r(\boldsymbol{x}) - c_{r-1}(\boldsymbol{x})$). The parameter vector θ is here given by $\theta = (\boldsymbol{w}, \beta_1, \ldots, \beta_{K-1})$. Note that, since the left-hand side in (20) is non-decreasing in k, the β_k need to satisfy the condition $\beta_1 \le \beta_2 \le \cdots \le \beta_{K-1}$.

5 Comparison of Reduction Techniques

Different reduction techniques are not easily comparable, especially because the performance of a meta-technique also depends on the base learner that is used to instantiate this technique. In this section, we nonetheless make an attempt at elaborating on commonalities and differences between existing reduction techniques (namely RPC and CC) and our new proposal (LWD), albeit not in much detail and not on a very technical level.

5.1 Complexity

If the original label ranking data consists of $|\mathbb{D}| = N$ complete examples, then the total number of examples generated by RPC is $NK(K-1)/2$. Since CC generates an example (actually even two) for each pairwise comparison, too, the same (or even twice this) number of examples can be produced for this method. However, whereas RPC distributes these examples over $K(K-1)/2$ instance spaces $\mathbb{X}_{i,j}$, which are all identical to the original space $\mathbb{X}$, CC combines them in a single expanded feature space $\overline{\mathbb{X}}$ whose dimensionality is K times as high, and solves a single problem in this space. In any case, even when leaving the dimensionality of the input space aside, RPC is theoretically more efficient than CC if the underlying base learner has a superlinear complexity, say, $O(N^\alpha)$ with $\alpha > 1$. In fact, in that case, solving $K(K-1)/2$ problems of size N is less expensive than solving a single problem of size $NK(K-1)/2$—the complexity of the former is $O(K(K-1)N^\alpha)$, while the latter is in $O((K(K-1)N)^\alpha)$.

It should also be mentioned that, in its original version, CC only constructs pairwise comparisons between *consecutive* labels in a ranking, not between all labels (hoping to capture the other relations implicitly via transitivity). In this case, the total number of examples reduces to $N(K-1)$. Of course, the same approach could be applied to any other pairwise method, including RPC. In terms of prediction performance, however, it turns out that the redundancy of the full encoding has significant advantages.

LWD constructs K classification problems of size N, thus KN examples in total; like in RPC, each of these problems uses the original input space $\mathbb{X}$. The complexity is not directly comparable, however, since LWD solves ordinal classification problems, whereas RPC and CC solve binary problems. Using decomposition techniques like those proposed by Frank and Hall [8], each ordinal problem could again be reduced to $K-1$ binary problems of the same size. Then, the overall complexity would be $O(K(K-1)N^\alpha)$, the same as for RPC.

Needless to say, a comparison becomes even more difficult in the case of incomplete training information. In that case, LWD requires methods for learning from imprecise data, such as (15). Therefore, the underlying base learners are no longer comparable.

In terms of space efficiency and complexity at prediction time, LWD may have an advantage in comparison to RPC, as it only needs to store and query a linear instead of a quadratic number of models. Again, however, since the LWD models are ordinal and the RPC models are binary classifiers, a direct comparison is not completely straightforward.

5.2 Loss of Information

Every reduction technique involves a certain loss of information. This can be seen most clearly from the fact that, from the information preserved on the level of the decomposition, the original probability distribution $\mathbf{P}(\cdot) = \mathbf{P}(\cdot \,|\, \boldsymbol{x})$ on $\mathbb{S}_K$ cannot always be recovered. For example, the uniform distribution $\mathbf{P}(\bar{\pi}) \equiv (K!)^{-1}$ and the bimodal distribution $\mathbf{P}'(\bar{\pi}) = 1/2$ for $\bar{\pi} = (1, 2, \ldots, K)$ and $\bar{\pi} = (K, K-1, \ldots, 1)$ (and $= 0$ otherwise) both induce the distribution $\mathbf{P}(y_i \succ y_j) \equiv 1/2$ on the level of pairwise comparisons. Thus, even if these pairwise probabilities were learned correctly, there is no chance to predict the true ranking from them. Obviously, the reason for this loss of information is the decomposition process itself: Decomposing a set of complex objects (in our case rankings) into a set of simple objects (e.g., pairwise preferences), the latter does not necessarily allow to recover the former.

As an important consequence, risk minimizing predictions cannot be produced for all loss functions. For example, as shown in [14], RPC is able to minimize (in expectation) the Kendall loss (2) and the Spearman loss (4) but not the L_1 loss (3). LWD, on the other hand, is able to minimize both L_1 and L_2, just like any other labelwise decomposable loss—this can be seen immediately from (12). It cannot minimize losses like Kendall, however, since probabilities of label inversions cannot be recovered from rank-probabilities on individual labels.

5.3 Modeling Incomplete Rank Information

As mentioned before, training information will normally not be provided in the form of complete rankings $\bar{\pi} \in \mathbb{S}_K$; instead, only incomplete examples (1) are available as training data. For a label ranking method, the ability to handle such information in a proper way is therefore of utmost importance.

Methods based on pairwise comparisons, such as CC and RPC, do have this ability and can handle missing label information in a quite straightforward way. In RPC, for example, if a label y_k is missing for a training instance $\boldsymbol{x}_n$, then none of the pairwise learners $\mathcal{M}_{i,j}$ with $k \in \{i, j\}$ will get $\boldsymbol{x}_n$ as an example. Similarly, missing labels reduce the number of training examples in CC. Yet, the examples that are produced from the observed labels are still *precise*. In other words, although missing labels reduce the number of examples, they do not affect the type and information content of those examples that are still produced. Correspondingly, the same learning algorithms can be used, and since they are applied to smaller data sets, the learning process will even become more efficient.

This is an important difference to LWD. Here, even a single missing label may affect all examples that are produced for a training instance $\boldsymbol{x}_n$—the class information (position of the label) will become imprecise and/or uncertain. Correspondingly, standard methods for ordinal classification are no longer applicable; instead, generalized methods for learning from imprecisely labeled examples must be used. Thus, missing label information may affect the quality of all examples that are derived from an instance $\boldsymbol{x}_n$ and, moreover, tend to increase the complexity of the learning problem instead of reducing it. Seen from this perspective, learning from *comparative* preferences does indeed appear to be advantageous to learning from *absolute* preferences.

6 Experiments

In this section, we experimentally compare LWD with RPC and CC in terms of prediction accuracy. All three meta-techniques are implemented using logistic regression as a base learner; RPC and CC get along with the basic binary version, whereas LWD requires an extended ordinal variant (cf. Section 4.4).

6.1 Data

We used several benchmark data sets for label ranking that have also been used in previous studies [15]; these are semi-synthetic data sets, namely label ranking versions of (real) UCI multi-class data. Moreover, we used two real label ranking data sets: The Sushi data[1] consists of 5000 instances (customers) described by 11 features, each one associated with a ranking of 10 types of sushis. The Students data [2] consists of 404 students (each characterized by 126 attributes) with associated rankings of five goals (want to get along with my parents, want to feel good about myself, want to have nice things, want to be different from others, want to be better than others). See Table 1 for a summary of the data.

Two missing label scenarios were simulated, namely the missing-at-random setting (16) and the top-rank setting (13). In the first case, a biased coin is flipped for every label in a ranking to decide whether to keep or delete that label; the probability for a deletion is specified by a parameter $p \in [0, 1]$. Thus, $p \times 100\%$ of the labels will be missing on average. Similarly, in the second case, only the J top-labels in a ranking are kept, where J has a binomial distribution with parameters K and $1 - p$.

Table 1: Properties of the data sets.

data set	# inst. (N)	# attr. (d)	# labels (K)
authorship	841	70	4
glass	214	9	6
iris	150	4	3
pendigits	10992	16	10
segment	2310	18	7
vehicle	846	18	4
vowel	528	10	11
wine	178	13	3
sushi	5000	11	10
students	404	126	5

6.2 Results

The results in Tables 2 and 3 are presented as averages of 5×10-fold cross validation in terms of the Kendall correlation measure; other measures such as (3) and (4) led to similar results. These tables support the following conclusions: (i) LWD and RPC perform much better than CC, which is not competitive. (ii) Overall, the drop in performance due to missing labels is more pronounced in the missing-at-random than in the top-rank setting. (iii) Compared with RPC, LWD is quite competitive if rankings are (almost) complete—in this case, it tends to be even a bit better; on the other hand, it drops in performance more quickly in the case of missing label information (the difference was found significant for 30% and 60% missing rate in the missing-at-random setting, using a two-tailed sign test at the 5% level).

7 Summary and Conclusion

In this paper, we introduced and analyzed labelwise decomposition (LWD) as a new meta-learning technique for label ranking. In contrast to existing techniques, which are

[1]http://kamishima.new/sushi/

Table 2: Performance in terms of Kendall's tau on synthetic data: missing-at-random (above) and top-rank setting (below).

	complete ranking			30% missing labels			60% missing labels		
	LWD	RPC	CC	LWD	RPC	CC	LWD	RPC	CC
authorship	.913±.01	.910±.02	.594±.04	.860±.02	.888±.03	.559±.05	.682±.02	.874±.03	.357±.06
glass	.883±.04	.882±.04	.834±.06	.837±.04	.854±.04	.825±.06	.760±.04	.790±.06	.748±.07
iris	.928±.06	.885±.07	.828±.06	.809±.06	.875±.07	.802±.07	.712±.08	.772±.10	.729±.11
pendigits	.928±.00	.932±.00	.584±.01	.914±.00	.932±.00	.534±.01	.895±.00	.929±.00	.506±.01
segment	.943±.01	.934±.01	.628±.05	.923±.01	.932±.01	.560±.06	.895±.01	.919±.01	.556±.10
vehicle	.867±.02	.854±.02	.839±.02	.828±.02	.834±.03	.823±.03	.759±.03	.778±.03	.759±.05
vowel	.674±.02	.647±.02	.577±.03	.656±.02	.643±.02	.548±.03	.609±.02	.612±.02	.525±.02
wine	.908±.06	.921±.05	.847±.10	.882±.06	.894±.07	.790±.07	.743±.07	.855±.10	.775±.12
Avg. Rank	1.25	1.75	3	1.875	1.125	3	2.375	1	2.625
authorship	.913±.01	.910±.02	.594±.04	.913±.02	.903±.02	.582±.04	.909±.02	.893±.03	.544±.04
glass	.883±.04	.882±.04	.834±.06	.872±.05	.880±.04	.824±.06	.812±.11	.845±.04	.819±.05
iris	.928±.06	.885±.07	.828±.06	.924±.05	.884±.07	.811±.07	.902±.09	.850±.09	.797±.06
pendigits	.928±.00	.932±.00	.584±.01	.919±.00	.931±.00	.535±.01	.863±.01	.920±.00	.507±.00
segment	.943±.01	.934±.01	.628±.05	.932±.01	.932±.01	.555±.07	.891±.03	.916±.01	.529±.12
vehicle	.867±.02	.854±.02	.839±.02	.859±.02	.850±.02	.828±.03	.841±.03	.832±.03	.812±.03
vowel	.674±.02	.647±.02	.577±.03	.665±.03	.645±.02	.567±.02	.619±.03	.645±.02	.527±.02
wine	.908±.06	.921±.05	.847±.10	.904±.05	.917±.06	.822±.10	.896±.07	.916±.05	.783±.10
Avg. Rank	1.25	1.75	3	1.5	1.5	3	1.75	1.375	2.875

Table 3: Performance in terms of Kendall's tau on real-world data: missing-at-random (above) and top-rank setting (below).

sushi	0%	10%	20%	30%	40%	50%	60%	70%
LWD	.329±.010	.328±.009	.329±.010	.328±.009	.328±.010	.327±.009	.325±.010	.321±.010
RPC	.329±.010	.329±.010	.328±.009	.328±.009	.327±.009	.327±.010	.325±.009	.322±.010
CC	.075±.011	.072±.012	.072±.011	.072±.013	.070±.013	.069±.012	.065±.012	.060±.013
LWD	.329±.010	.329±.010	.329±.010	.329±.010	.328±.010	.325±.010	.323±.010	.319±.010
RPC	.329±.010	.329±.010	.329±.010	.329±.010	.328±.010	.326±.010	.324±.010	.321±.010
CC	.075±.011	.069±.013	.071±.012	.071±.013	.072±.012	.069±.012	.068±.011	.065±.009

students	0%	10%	20%	30%	40%	50%	60%	70%
LWD	.500±.046	.474±.053	.459±.055	.431±.050	.411±.054	.391±.054	.376±.059	.389±.066
RPC	.477±.037	.471±.052	.458±.052	.458±.056	.443±.063	.445±.044	.446±.052	.445±.045
CC	.455±.064	.424±.068	.339±.073	.304±.056	.316±.062	.284±.058	.274±.064	.268±.058
LWD	.500±.046	.497±.048	.499±.044	.496±.044	.481±.048	.451±.042	.420±.057	.397±.056
RPC	.477±.056	.460±.053	.456±.056	.452±.059	.445±.058	.441±.058	.449±.052	.445±.048
CC	.455±.064	.457±.067	.448±.069	.438±.064	.378±.065	.378±.058	.268±.072	.162±.073

based on decomposing training information into *comparative* preferences, this approach is based on *absolute* preference information in the form of ranks. The idea is quite simple: For each individual label, a model is learned that, given a query instance as an input, predicts the rank of the label in the associated ranking.

Technically, LWD reduces label ranking to ordinal classification problems with imprecise class information. Moreover, the aggregation step, which is responsible for combining the predictions of these classifiers into a complete label ranking, can be realized by means of an optimal assignment problem—this way, each labelwise decomposable loss function can be minimized in expectation.

Comparing LWD with state-of-the-art reduction techniques for label ranking, we did not find any systematic improvements in terms of prediction accuracy. On the contrary, although improvements could be achieved on several data sets in the case of (almost) complete training data, LWD seems to be more sensitive to missing label information. Actually, these results fully confirm our expectations, and can be explained by the fact that absolute preference information is more strongly affected by missing labels than relative preference information.

Overall, however, and especially in light of the unambitious expectations we started with, we found LWD to be surprisingly competitive. Moreover, one should keep in mind that LWD is a meta-learning technique whose performance is strongly influenced by the base learner. Since the implementation of this base learner is non-trivial, and the version used in this paper not necessarily optimal, there is certainly scope to improve this part of the method. Besides, LWD has other interesting properties. For example, while its performance is competitive to RPC, it only needs a linear instead of a quadratic number of models, which might not only be advantageous from a complexity point of view but also interesting with regard to the comprehensibility of a label ranker.

All things considered, we therefore believe that our results, despite not (yet) advancing the state-of-the-art in terms of performance, are promising enough to justify a further investigation of LWD as an alternative learning technique for label ranking. For future work, we therefore plan to explore this approach in more depth and to develop it further, with the goal to fully exploit its potential.

References

[1] G. Bakir, T. Hofmann, B. Schölkopf, A. Smola, B. Taskar, and S. Vishwanathan, editors. *Predicting structured data*. MIT Press, 2007.

[2] M. Boekaerts, K. Smit, and F.M.T.A. Busing. Salient goals direct and energise students' actions in the classroom. *Applied Psychology: An International Review*, 4(S1):520–539, 2012.

[3] K. Brinker and E. Hüllermeier. Case-based label ranking. In *Proceedings ECML–06, 17th European Conference on Machine Learning*, pages 566–573, Berlin, September 2006. Springer-Verlag.

[4] R.E. Burkard, M. Dell'Amico, and S. Martello. *Assignment Problems*. SIAM, 2009.

[5] W. Cheng, K. Dembczynski, and E. Hüllermeier. Label ranking based on the Plackett-Luce model. In

J. Fürnkranz and T. Joachims, editors, *Proceedings ICML–2010, International Conference on Machine Learning*, Haifa, Israel, 2010.

[6] W. Cheng, J. Hühn, and E. Hüllermeier. Decision tree and instance-based learning for label ranking. In *Proceedings ICML–2009, 26th International Conference on Machine Learning*, Montreal, Canada, 2009. [27% acceptance rate].

[7] T. Cour, B. Sapp, and B. Taskar. Learning from partial labels. *Journal of Machine Learning Research*, 12:1501–1536, 2011.

[8] E. Frank and M. Hall. A simple approach to ordinal classification. In *Proc. ECML–2001, 12th European Conference on Machine Learning*, pages 145–156, Freiburg, Germany, 2001.

[9] J. Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2:721–747, 2002.

[10] J. Fürnkranz and E. Hüllermeier, editors. *Preference Learning*. Springer-Verlag, 2011.

[11] Y. Grandvalet. Logistic regression for partial labels. In *IPMU–02, Int. Conf. Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 1935–1941, Annecy, France, 2002.

[12] 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.

[13] E. Hüllermeier and J. Beringer. Learning from ambiguously labeled examples. *Intelligent Data Analysis*, 10(5):419–440, 2006.

[14] E. Hüllermeier and J. Fürnkranz. On predictive accuracy and risk minimization in pairwise label ranking. *Journal of Computer and System Sciences*, 76(1):49–62, 2010.

[15] E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172:1897–1917, 2008.

[16] H.W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, 1955.

[17] G. Tsoumakas and I. Katakis. Multi-label classification: An overview. *Int. J. of Data Warehousing and Mining*, 3(3):1–13, 2007.

[18] S. Vembu and T. Gärtner. Label ranking: a survey. In J. Fürnkranz and E. Hüllermeier, editors, *Preference Learning*. Springer-Verlag, 2010.