# Dyad Ranking Using A Bilinear Plackett-Luce Model

Dirk Schäfer[1] and Eyke Hüllermeier[2](✉)

[1] University of Marburg, Marburg, Germany
dirk.schaefer@uni-marburg.de
[2] Department of Computer Science, University of Paderborn, Paderborn, Germany
eyke@upb.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. These alternatives are identified by their name or *label* while not being characterized in terms of any properties or features that could be potentially useful for learning. In this paper, we consider a generalization of the label ranking problem that we call *dyad ranking*. In dyad ranking, not only the instances but also the alternatives are represented in terms of attributes. For learning in the setting of dyad ranking, we propose an extension of an existing label ranking method based on the Plackett-Luce model, a statistical model for rank data. Moreover, we present first experimental results confirming the usefulness of the additional information provided by the feature description of alternatives.

**Keywords:** Label ranking · Plackett-Luce model · Meta-learning

## 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 [7]. 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 [26]. An instance, which defines the context of the preference relation, is typically characterized in terms of a set of attributes or features; for example, an instance could be a person described by properties such as sex, age, income, etc. As opposed to this, the alternatives to be ranked, e.g., the political parties of a country, are only identified by their name (label), while not being characterized in terms of any properties or features.


A. Appice et al. (Eds.): ECML PKDD 2015, Part II, LNAI 9285, pp. 227–242, 2015.
DOI: 10.1007/978-3-319-23525-7_14

In this paper, we introduce *dyad ranking* as a generalization of the label ranking problem. In dyad ranking, not only the instances but also the alternatives are represented in terms of attributes. For learning in the setting of dyad ranking, we propose an extension of an existing label ranking method based on the Plackett-Luce model, a statistical model for rank data.

The paper is organized as follows. In the next section, we introduce the problem of dyad ranking. Following a discussion of related work in Section 3, we then propose the aforementioned method for dyad ranking in Section 4. In Section 5, we present first experimental results, both for synthetic data and a case study in meta-learning, confirming the usefulness of the additional feature information of alternatives. The paper ends with some concluding remarks in Section 6.

## 2 Dyad Ranking

As will be explained in more detail later on (cf. Section 3), the learning problem addressed in this paper has connections to several existing problems in the realm of preference learning. In particular, it can be seen as a combination of *dyadic prediction* [19–21] and *label ranking* [26], hence the term "dyad ranking". Since our method for tackling this problem is an extension of a label ranking method, we will introduce dyad ranking here as an extension of label ranking.

### 2.1 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* or simply *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 $y_j$. We interpret this order relation in a wide sense, so that $a \succ b$ can mean that the alternative $a$ is more liked that alternative $b$ by a person, but also for example that an algorithm $a$ outperforms algorithm $b$.

Formally, a total order $\succ$ can be identified with a permutation $\pi$ of the set $[K] = \{1, \ldots, K\}$, such that $\pi(i)$ is the index of the label on position $i$. 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 $\pi \in \mathbb{S}_K$ 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 $\pi_{\boldsymbol{x}} \in \mathbb{S}_K$. 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 $\pi \in \mathbb{S}_K$, $\mathbf{P}(\pi \,|\, \boldsymbol{x})$ denotes the probability to observe the ranking $\pi$ 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 $\pi = (2, 1, 3, 4)$, $\mathbf{P}(\pi \,|\, \boldsymbol{x})$ denotes the probability that a 30 years old married man, who is a teacher, prefers Pop music to Rock to Classic to Jazz.

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 $\pi$ 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

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

where $\mathbf{M}$ is the underlying model class, $\mathbf{P}$ is the joint measure $\mathbf{P}(\boldsymbol{x}, \pi) = \mathbf{P}(\boldsymbol{x})\mathbf{P}(\pi \,|\, \boldsymbol{x})$ on $\mathbb{X} \times \mathbb{S}_K$ and $\mathcal{L}$ is a loss function on $\mathbb{S}_K$.

As training data $\mathcal{D}$, a label ranker uses a set of instances $\boldsymbol{x}_n$ $(n \in [N])$, together with information about the associated rankings $\pi_n$. Ideally, complete rankings are given as training information, i.e., a single observation is a tuple of the form $(\boldsymbol{x}_n, \pi_n) \in \mathbb{X} \times \mathbb{S}_K$. 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 $\mathcal{Y}$:

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

where $J \leq K$ and $\{\pi(1), \ldots, \pi(J)\} \subset [K]$. 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 $y_4$.

### 2.2 Dyad Ranking as an Extension of Label Ranking

In the setting of label ranking as introduced above, instances are supposed to be characterized in terms of properties—typically, an instance is represented as an $r$-dimensional feature vector $\boldsymbol{x} = (x_1, \ldots, x_r)$. As opposed to this, the alternatives to be ranked, the labels $y_i$, are only identified by their name, just like categories in classification.

Needless to say, a learner may benefit from knowledge about properties of the alternatives, too. In fact, if the preferences of an instance are somehow connected to such properties, then alternatives with similar properties should also be ranked similarly. In particular, by sharing information via features, it would in principle be possible to rank alternatives that have never be seen in the training process so far.

Returning to our above example of ranking music genres, suppose we know (or at least are quite sure) that $\texttt{Rock} \succ_{\boldsymbol{x}} \texttt{Classic} \succ_{\boldsymbol{x}} \texttt{Jazz}$ for a person $\boldsymbol{x}$. We would then expect that `Pop` is ranked more likely close to the top than

close to the bottom, simply because Pop music is more similar to Rock than to Classic or Jazz. In contrast to a label ranker, for which the music genres are just uninformative names, we are able to make a prediction of that kind thanks to our knowledge about the different types of music.

Given that useful properties of alternatives are indeed often available in practice, we introduce dyad ranking as an extension of label ranking, in which alternatives are elements of a feature space:

$$\boldsymbol{y} = (y_1, y_2, \ldots, y_c) \in \mathbb{Y} = \mathbb{Y}_1 \times \mathbb{Y}_2 \times \cdots \times \mathbb{Y}_c \tag{2}$$

Then, a *dyad* is a pair

$$\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{Z} = \mathbb{X} \times \mathbb{Y} \tag{3}$$

consisting of an instance $\boldsymbol{x}$ and an alternative $\boldsymbol{y}$. We assume training information to be given in the form of rankings

$$\rho_i : \boldsymbol{z}^{(1)} \succ \boldsymbol{z}^{(2)} \succ \ldots \succ \boldsymbol{z}^{(M_i)} \tag{4}$$

of a finite number of dyads, where $M_i$ is the length of the ranking. Typically, though not necessarily, all dyads in (11) share the same context $\boldsymbol{x}$, i.e., they are all of the form $\boldsymbol{z}^{(j)} = (\boldsymbol{x}, \boldsymbol{y}^{(j)})$; in this case, (11) can also be written

$$\rho_i : \boldsymbol{y}^{(1)} \succ_{\boldsymbol{x}} \boldsymbol{y}^{(2)} \succ_{\boldsymbol{x}} \ldots \succ_{\boldsymbol{x}} \boldsymbol{y}^{(M_i)} \ . \tag{5}$$

Likewise, a prediction problem will typically consist of ranking a subset

$$\left\{ \boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(M)} \right\} \subseteq \mathbb{Y}$$

in a given context $\boldsymbol{x}$. Given a *dyad ranker*, i.e., a model that produces a ranking of dyads as an output, this can be accomplished by applying that ranker to the set of dyads

$$\left(\boldsymbol{x}, \boldsymbol{y}^{(1)}\right), \left(\boldsymbol{x}, \boldsymbol{y}^{(2)}\right), \ldots, \left(\boldsymbol{x}, \boldsymbol{y}^{(M)}\right)$$

and then projecting the result to the alternatives, i.e., transforming a ranking of the form (11) into one of the form (5). This setting, which generalizes label ranking in the sense that additional information in the form of feature vectors is provided for the labels, is the main subject of this paper and will subsequently be referred to as *contextual dyad ranking*.

## 3 Related Work

As already mentioned earlier, the problem of dyad ranking is not only connected to label ranking, but also to several other types of ranking and preference learning problems that have been discussed in the literature. Although a comprehensive review of related work is beyond the scope of this paper, we shall give a brief overview in this section.

The term "dyad ranking" derives from the framework of *dyadic prediction* as introduced by Menon and Elkan [20]. This framework can be seen as a generalization of the setting of collaborative filtering (CF), in which *row-objects* (e.g., clients) are distinguished from *column-objects* (e.g., products). Moreover, with each combination of such objects, called a dyad by Menon and Elkan, a value (e.g., a rating) is associated. While in CF, row-objects and column-objects are only represented by their name (just like the alternatives in label ranking), they are allowed to have a feature representation (called side-information) in dyadic prediction. Menon and Elkan are trying to exploit this information to improve performance in matrix completion, i.e., predicting the values for those object combinations that have not been observed so far, in very much the same way as we are trying to make use of feature information in the context of label ranking.

Methods for *learning-to-rank* or *object ranking* [6,13] have received a lot of attention in the recent years, especially in the field of information retrieval. In general, the goal is to learn a *ranking function* that accepts a subset $\mathbf{O} \subset \mathbb{O}$ of objects as input, where $\mathbb{O}$ is a reference set of objects (e.g., the set of all books). As output, the function produces a ranking (total order) $\succ$ of the objects $\mathbf{O}$. The ranking function is commonly implemented by means of a scoring function $U : \mathbb{O} \to \mathbb{R}$, i.e., objects are first scored and then ranked according to their scores. In order to induce a function of that kind, the learning algorithm is provided with training information, which typically comes in the form of exemplary pairwise preferences between objects. As opposed to label ranking, the alternatives to be ranked are described in terms of properties (feature vectors), while preferences are not contextualized. In principle, methods for object ranking could be applied in the context of dyad ranking, too, namely by equating the object space $\mathbb{O}$ with the "dyad space" $\mathbb{Z}$ in (3); in fact, dyads can be seen as a specific type of object, i.e., as objects with a specific structure. Especially close in terms of the underlying methodology is the so-called *listwise approach* in learning-to-rank [4].

Close to our setting is also the (kernel-based) framework of *conditional ranking* [24]. Here, relational data is represented in terms of a graph structure, in which nodes correspond to objects and (directed) edges are labeled with associations between these objects. Conditional ranking then refers to the problem of ranking a set of nodes relative to another (target) node, namely, of ranking the former in decreasing order of association with the latter. Associations are modeled in terms of a specific type of kernel function called *preference kernel*, and an SVM-like training procedure (with quadratic instead of hinge loss) is used for model induction. The framework is quite flexible and covers different learning problems as special cases, depending on the type of graph (bipartite or complete), the type of edge labels and the type of training information [23].

## 4 A Bilinear Plackett-Luce Model

### 4.1 The Plackett-Luce Model

The Plackett-Luce (PL) model is a parameterized probability distribution on the set of all rankings over a set of alternatives $y_1, \ldots, y_K$. It is specified by a

parameter vector $\boldsymbol{v} = (v_1, v_2, \dots v_K) \in \mathbb{R}_+^K$, in which $v_i$ accounts for the "skill" of the option $y_i$. The probability assigned by the PL model to a ranking represented by a permutation $\pi$ is given by

$$\mathbf{P}(\pi \mid \boldsymbol{v}) \;=\; \prod_{i=1}^{K} \frac{v_{\pi(i)}}{v_{\pi(i)} + v_{\pi(i+1)} + \ldots + v_{\pi(K)}} \tag{6}$$

This model is a generalization of the well-known Bradley-Terry model [18], a model for the pairwise comparison of alternatives, which specifies the probability that "$a$ wins against $b$" in terms of

$$\mathbf{P}(a \succ b) \;=\; \frac{v_a}{v_a + v_b} \;\; .$$

Obviously, the larger $v_a$ in comparison to $v_b$, the higher the probability that $a$ is chosen. Likewise, the larger the parameter $v_i$ in (6) in comparison to the parameters $v_j$, $j \neq i$, the higher the probability that $y_i$ appears on a top rank.

As a nice feature of Plackett-Luce, we note that marginals (i.e., probabilities of rankings of a subset of the alternatives) can be computed very easily for this model: The probability of an incomplete ranking (1) is given by

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

i.e., by an expression of exactly the same form as (6), except that the number of factors is $J$ instead of $K$.

### 4.2 Label Ranking Using the PL Model

A method for label ranking based on the PL model was proposed in [5]. The main idea of this approach is to contextualize the skill parameters of the labels $y_i$ by modeling them as functions of the context $\boldsymbol{x}$. More precisely, to guarantee the non-negativity of the parameters, they are modeled as log-linear functions:

$$v_k = v_k(\boldsymbol{x}) = \exp\left(\sum_{d=1}^{r} w_d^{(k)} \cdot x_d\right) = \exp\left(\left\langle \boldsymbol{w}^{(k)}, \boldsymbol{x} \right\rangle\right) \;\; . \tag{7}$$

The parameters of the label ranking model, namely the $w_d^{(k)}$ $(1 \leq k \leq K,\, 1 \leq d \leq r)$, are estimated by maximum likelihood inference.

Given estimates of these parameters, prediction for new query instances $\boldsymbol{x}$ can be done in a straightforward way: $\hat{\boldsymbol{v}} = (\hat{v}_1, \ldots, \hat{v}_K)$ is computed based on (7), and a ranking $\hat{\pi}$ is determined by sorting the labels $y_k$ in decreasing order of their (predicted) skills $\hat{v}_k$. This ranking $\hat{\pi}$ is a reasonable prediction, as it corresponds to the mode of the distribution $\mathbf{P}(\cdot \mid \hat{\boldsymbol{v}})$.

### 4.3 Dyad Ranking Using the PL model

In (7), the skill of the label $y_k$ is modeled as a log-linear function of $\boldsymbol{x}$, with a label-specific weight vector $\boldsymbol{w}^{(k)}$. In the context of dyad ranking, this approach can be generalized to the modeling of skills for dyads as follows:

$$v(\boldsymbol{z}) = v(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(\langle \boldsymbol{w}, \Phi(\boldsymbol{x}, \boldsymbol{y})\rangle\right) \ , \tag{8}$$

where $\Phi$ is a joint feature map [25]. A common choice for such a feature map is the Kronecker product:

$$\Phi(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x} \otimes \boldsymbol{y} = \left(x_1 \cdot y_1, x_1 \cdot y_2, \ldots, x_r \cdot y_c\right) = vec\left(\boldsymbol{x}\boldsymbol{y}^\top\right) \ , \tag{9}$$

which is a vector of length $r \cdot c$ consisting of all pairwise products of the components of $\boldsymbol{x}$ and $\boldsymbol{y}$. Thus, the inner product $\langle \boldsymbol{w}, \Phi(\boldsymbol{x}, \boldsymbol{y})\rangle$ can be rewritten as a bilinear form $\boldsymbol{x}^\top \mathbf{W} \boldsymbol{y}$ with an $r \times c$ matrix $\mathbf{W} = (w_{i,j})$; the entry $w_{i,j}$ can be considered as the weight of the interaction term $x_i y_j$. This choice of the joint-feature map yields a bilinear version of the PL model:

$$v(\boldsymbol{z}) = v(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(\boldsymbol{x}^\top \mathbf{W} \boldsymbol{y}\right) \tag{10}$$

Suppose training data $\mathcal{D}$ to be given in the form of a set of rankings (11), i.e., rankings $\rho_1, \ldots, \rho_N$ of the following kind:

$$\rho_n : (\boldsymbol{x}_n^{(1)}, \boldsymbol{y}_n^{(1)}) \succ (\boldsymbol{x}_n^{(2)}, \boldsymbol{y}_n^{(2)}) \succ \ldots \succ (\boldsymbol{x}_n^{(M_n)}, \boldsymbol{y}_n^{(M_n)}) \tag{11}$$

The likelihood of the parameter vector $\boldsymbol{w}$ is then given by

$$L(\boldsymbol{w}) = \mathbf{P}(\mathcal{D} \,|\, \boldsymbol{w}) = \prod_{n=1}^{N} \prod_{m=1}^{M_n} \frac{\exp\left(\boldsymbol{w}^\top (\boldsymbol{x}_n^{(m)} \otimes \boldsymbol{y}_n^{(m)})\right)}{\sum_{l=m}^{M_n} \exp\left(\boldsymbol{w}^\top (\boldsymbol{x}_n^{(l)} \otimes \boldsymbol{y}_n^{(l)})\right)} \ ,$$

and the log-likelihood by

$$\ell(\boldsymbol{w}) = \sum_{n=1}^{N} \sum_{m=1}^{M_n} \boldsymbol{w}^\top (\boldsymbol{x}_n^{(m)} \otimes \boldsymbol{y}_n^{(m)}) - \sum_{n=1}^{N} \sum_{m=1}^{M_n} \log\left(\sum_{l=m}^{M_n} \exp\left(\boldsymbol{w}^\top (\boldsymbol{x}_n^{(l)} \otimes \boldsymbol{y}_n^{(l)})\right)\right).$$

Like in the case of the linear PL model, the learning problem can now be formalized as finding the maximum likelihood (ML) estimate, i.e., the parameter $\boldsymbol{w}$ that maximizes the log-likelihood:

$$\boldsymbol{w}_{ML} = \underset{\boldsymbol{w}}{\operatorname{argmax}}\ \ell(\boldsymbol{w}) \ , \tag{12}$$

To save the costly computations of the Hessian during ML estimation, a quasi-Newton type algorithm (L-BFGS, [17]) is used in our implementation. Further remarks on the identifiability of the model parameters are provided below.

### 4.4 Identifiability of the Bilinear PL Model

The bilinear PL model introduced above defines a probability distribution on dyad rankings that is parameterized by the weight matrix $\mathbf{W}$. An interesting question concerns the identifiability of this model. Recall that, for a parameterized class of models $\mathcal{M}$, identifiability requires a bijective relationship between models $M_\theta \in \mathcal{M}$ and parameters $\theta$, that is, models are uniquely identified by their parameters. Or, stated differently, parameters $\theta \neq \theta^*$ induce different models $M_\theta \neq M_{\theta^*}$. Identifiability is a prerequisite for a meaningful interpretation of parameters and, perhaps even more importantly, guarantees unique solutions for optimization procedures such as maximum likelihood estimation.

Obviously, the original PL model (6) with constant skill parameters $\boldsymbol{v} = (v_1, \ldots, v_K)$ is not identifiable, since the model is invariant against multiplication of the parameter by a constant factor $c > 0$: The models parameterized by $\boldsymbol{v}$ and $\boldsymbol{v}^* = (cv_1, \ldots, cv_K)$ represent exactly the same probability distribution, i.e., $\mathbf{P}(\pi \,|\, \boldsymbol{v}) = \mathbf{P}(\pi \,|\, \boldsymbol{v}^*)$ for all rankings $\pi$. The PL model is, however, indeed identifiable up to this kind of multiplicative scaling. Thus, by fixing one of the weights to the value 1, the remaining $K - 1$ weights can be uniquely identified.

Now, what about the identifiability of our bilinear PL model, i.e., to what extent is such a model uniquely identified by the parameter $\mathbf{W}$? We can show the following result.

**Proposition 1:** *Suppose the feature representation of labels does not include a constant feature, i.e., $|\mathbb{Y}_i| > 1$ for each of the domains in (2), and that the feature representation of instances includes at most one such feature (accounting for a bias, i.e., an intercept of the bilinear model). Then, the bilinear PL model with skill values defined according to (10) is identifiable.*

Proof (sketch): Recall that the standard PL model is invariant against multiplication with a positive constant, and that this is the only invariance of the model. Since the bilinear PL model defined by (10) is log-linear in $\mathbf{W}$, invariance on the level of this parameter can only be additive. Now, suppose there are two parameters $\mathbf{W} \neq \mathbf{W}^*$ that both induce the same distribution on the set of all potential dyad subsets, which means that

$$\boldsymbol{x}^\top \mathbf{W} \boldsymbol{y} = \boldsymbol{x}^\top \mathbf{W}^* \boldsymbol{y} + \gamma \tag{13}$$

for all dyads $(\boldsymbol{x}, \boldsymbol{y})$, where $\gamma$ is a constant that may depend on the parameters $\mathbf{W}$ and $\mathbf{W}^*$ but *not* on the dyads $(\boldsymbol{x}, \boldsymbol{y})$. More specifically, for the case of contextual dyad ranking, $\gamma$ is also allowed to depend on $\boldsymbol{x}$, but again, must not depend on $\boldsymbol{y}$. Under our assumptions, however, this independence cannot hold. In fact, denoting the elements of $\mathbf{W}$ and $\mathbf{W}^*$ by $w_{i,j}$ and $w^*_{i,j}$, respectively, (13) means that

$$\sum_{i=1}^{r} \sum_{j=1}^{c} (w_{i,j} - w^*_{i,j}) x_i y_j = \sum_{i=1}^{r} \sum_{j=1}^{c} \Delta w_{i,j} x_i y_j = \gamma \ .$$

Then, exploiting the fact that not all $\Delta w_{i,j}$ can vanish at the same time, it is not difficult to show that a variation of some values $y_j$, which will also have an influence on the difference $\gamma$, is always possible.

### 4.5 Comparison Between the Linear and Bilinear PL Model

It is not difficult to see that the linear model (7), subsequently referred to as LinPL, is indeed a special case of the bilinear model (10), called BilinPL. In fact, the former is recovered from the latter by means of a (1-of-$K$) dummy encoding of the alternatives: The label $y_k$ is encoded by a $K$-dimensional vector with a 1 in position $k$ and 0 in all other positions. The columns of the matrix $\mathbf{W}$ are then given by the weight vectors $\boldsymbol{w}^{(k)}$ in (7).

The other way around, LinPL can also be applied in the setting of dyad ranking, provided the domain $\mathbb{Y}$ of the alternatives is finite. To this end, one would simply introduce one "meta-label" $Y_k$ for each feature combination $(y_1, \ldots, y_c)$ in (2) and apply a standard label ranking method to the set of these meta-labels. Therefore, both approaches are in principle equally expressive. Still, an obvious problem of this transformation is the potential size[1]

$$K = |\mathbb{Y}| = |\mathbb{Y}_1| \times |\mathbb{Y}_2| \times \ldots \times |\mathbb{Y}_c|$$

of the label set thus produced, which might be huge. In fact, the number of parameters that need to be learned for the model (7) is $r \cdot |\mathbb{Y}|$, i.e., $r \cdot a^c$ under the assumptions that each feature has $a$ values. For comparison, the number of parameters is only $r \cdot c$ in the bilinear model. Moreover, all information about relationships between the alternatives (such as shared features or similarities) are lost, since a standard label ranker will only use the name of a meta-label while ignoring its properties.

Against the background of these consideration, one should expect dyad ranking to be advantageous to standard label ranking provided the assumptions underlying the bilinear model (10) are indeed valid, at least approximately. In that case, learning with (meta-)labels and disregarding properties of the alternatives would come with an unnecessary loss of information (that would need to be compensated by additional training data). In particular, using the standard label ranking approach is supposedly problematic in the case of many meta-labels and comparatively small amounts of training data.

Having said that, dyad ranking could be problematic if the model (10) is in fact a misspecification: If the features are not meaningful, or the bilinear model is not properly reflecting their interaction, then learning on the basis of (10) cannot be successful.

In this regard, it is also interesting to mention that both approaches can be combined. To this end, the feature vectors $\boldsymbol{y}$ are extended by a (1-of-$K$) dummy-encoding, i.e., dyad ranking is used with feature vectors of the following form:

$$\boldsymbol{y} = \big( y_1, y_2, \ldots, y_c, \underbrace{0, \ldots, 0, 1, 0, \ldots, 0}_{\text{length } K} \big) \tag{14}$$

[1] This is an upper bound, since in practice, not all feature combinations are necessarily realized.

Using this representation, subsequently called LinSidePL, the learner is in principle free to exploit the side-information $y_i$ or to ignore it and only use the dummy-labels.

In summary, the main observations can be summarized as follows:

- The linear PL model, like standard label ranking in general, assumes all alternatives to be known beforehand and to be included in the training process. If generalization beyond alternatives encountered in the training process is needed, then BilinPL can be used while LinPL cannot.
- If the assumption (10) of the bilinear model is correct, then BilinPL should learn faster than LinPL, as it needs to estimate fewer parameters. Yet, since LinPL can represent all dependencies that can be represented by BilinPL, the learning curve of the former should reach the one of latter with growing sample size.
- If the bilinear model (10) is actually a misspecification, then LinPL is likely to perform better than BilinPL, at least with enough training data being available (for small training sets, BilinPL could still be better).

## 5 Experiments

In order to verify the expectations summarized above, we conducted experiments with both synthetic and real data sets. In addition to LinPL (as implemented in [5]), BilinPL and LinSidePL, we included Ranking by Pairwise Comparison (RPC, [12]) and Constrained Classification (CC, [9,10]), which are both state-of-the-art label ranking methods, as additional baselines.[2]

Predictive performance was measured in terms of Kendall's tau coefficient [15], a rank correlation measure commonly used for this purpose in the label ranking literature [26,27]. It is defined as

$$\tau = \frac{C(\pi, \hat{\pi}) - D(\pi, \hat{\pi})}{K(K-1)/2} \quad , \tag{15}$$

with $C$ and $D$ the number of concordant (put in the same order) and discordant (put in the reverse order) label pairs, respectively, and $K$ the length of the rankings $\pi$ and $\hat{\pi}$ (number of labels). Kendall's tau assumes values in $[-1, +1]$, with $\tau = +1$ for the perfect prediction $\hat{\pi} = \pi$ and $\tau = -1$ if $\hat{\pi}$ is the exact reversal of $\pi$.

### 5.1 Synthetic Data

Ideal synthetic ranking data is created by sampling from the Plackett-Luce distribution according to the BilinPL model specification under the setting (5) of contextual dyad ranking. A realistic scenario is simulated in which labels can be missing, i.e., observed rankings are incomplete [11]. To this end, a biased coin is

[2] CC was used in its online variant as described in [12].

flipped for every label, and it is decided with probability $p \in [0, 1]$ to keep or to delete it. We choose a missing rate of $p = 0.3$, which means that on average 70% of all labels of the training set are kept while the remaining labels are dismissed. Feature vectors of length $c = 4$ for labels and length $r = 3$ for instances were generated by sampling the elements from a standard normal distribution (except for one instance feature, which is a constant). The weight components were sampled randomly from a normal distribution with mean 1 and standard deviation 9. The predictive performance is then determined on a sufficiently large number of (complete) test examples and averaged over 10 repetitions.

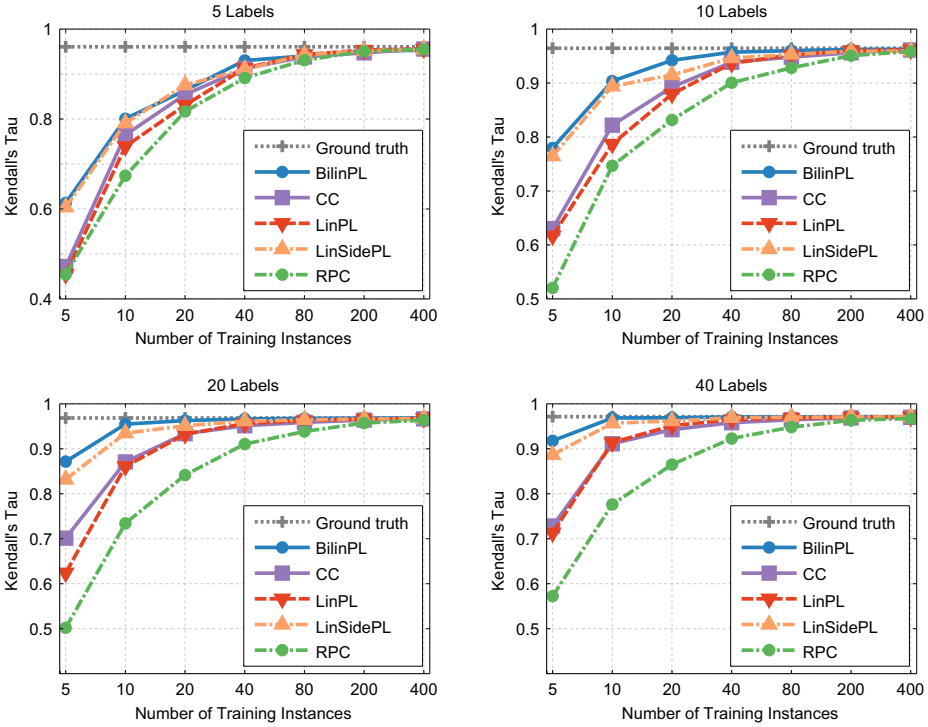


**Fig. 1.** Learning curves (generalization performance as a function of the number of training examples) of the ranking methods for different numbers of labels.

The learning curves thus produced are shown in Figure 1 for different numbers of labels. Overall, all ranking methods are able to learn and predict correctly if enough training data are available. In the limit, they all reach the performance of the "ground truth": given complete knowledge about the true PL model, the optimal (Bayes) prediction is the mode of that distribution (note that the average performance of that predictor is still not perfect, since sampling from the distribution will not always yield the mode). As expected, BilinPL and LinSidePL both benefit from the additional label description compared to the other

label ranking approaches over a wide range of different training set sizes and numbers of labels.

Apart from predictive accuracy, it is worth mentioning that, in comparison with the BilinPL model, standard label ranking methods also exhibit poor runtime characteristics.

### 5.2 Case Study in Meta-Learning

As conjectured in Section 4.5 and confirmed in Section 5.1, BilinPL is potentially advantageous to LinPL in cases where the number of alternatives (labels) is large in comparison to the amount of training information being available and, moreover, these alternatives can be described in terms of suitable features. An interesting application for which these assumptions seem to hold is meta-learning [2]. In this section, we therefore employ the framework of meta-learning for algorithm recommendation as described in [2,3,14]. In particular, we aim at predicting a ranking over several variants of a class of algorithms such as genetic algorithms (GA), which can be obtained by instantiating the algorithm with different parameter combinations.

Several choices need to be made within the meta-learning framework, including the way of how meta-data is acquired (see Figure 2). The meta-features as part of the meta-data should be able to relate a data set to the relative performance of the candidate algorithms. They are usually made up by a set of numbers acquired by using descriptive statistics. Another possibility consists of probing a few parameter settings of the algorithm under consideration. The performance values of those *landmarkers* can then be used as instance-features for the meta-learner. In addition to the meta-features, the meta-data consists of rankings of the candidate algorithms, i.e., a sorting of the variants in decreasing order of performance. Using the meta-learning terminology, these rankings correspond to the so-called meta-target. The novel aspect in this paper is the use of qualitative performance data in the form of rankings[3] in conjunction with the consideration of side-information.

In analogy to the majority classifier typically used as a baseline in multi-class classification, the meta-learning literature suggests a simple approach called the Average Ranks (AR) method [2]. This approach corresponds to what is called the Borda count in the ranking literature and produces a default prediction by sorting the alternatives according to their average position in the observed rankings.

**Learning to Rank Genetic Algorithms.** This case study aims at recommending GA parameter settings for instances of the symmetric traveling salesman problem (TSP). The GA performance averages are taken to construct rankings, in which a single performance value corresponds to the distance of the shortest route found by a GA. The GAs share the properties of using the same

[3] This also comprises partial rankings and pairwise preferences as special cases.

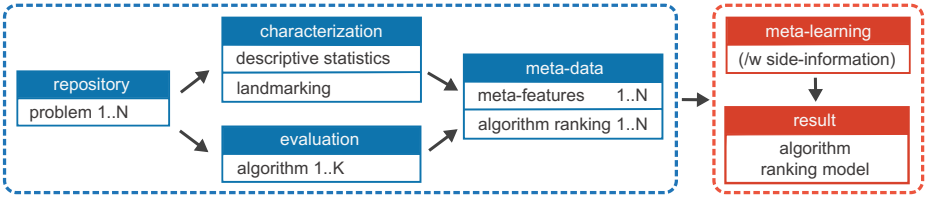


**Fig. 2.** The components of the "meta-learning for algorithm recommendation" framework shown above are based on [2]. The left box shows the meta-data acquisition process which consists of learning problem (or data set) characterization and the evaluation of the algorithms on the problems (or data sets). The box on the right side, the meta-level learning part, shows the meta-learning process and its outcome. In this case study, the meta-learner must be able to deal with qualitative data in form of rankings and is furthermore allowed to use additional knowledge (side-information) about the algorithms if it is available.

selection criterion, which is "roulette-wheel", the same mutation operator, which is "exchange mutation" and "elitism" of 10 chromosomes [22]. We tested the performance of three groups of GAs on a set of TSP instances. The groups are determined by their choice of the crossover operator, which can be cycle (CX), order (OX) or partially mapped crossover (PMX) [16].

The set of meta-features represent the instance vectors for the ranking models. They are composed of the number of cities and the performances of three landmarkers.

In total, 246 problems are considered, with the number of cities ranging between 10 and 255.[4] For each problem, the city locations $(x, y)$ are drawn randomly from the uniform distribution on $[1, 100]^2$. Moreover, 72 different GAs are considered as alternatives with their parameters as optional label descriptions. They share the number of generations, 500, and the population size of 100. The combinations of all the other parameters, namely, crossover type, crossover rate and mutation rate, are used for characterization:

- Crossover types: {CX, OX, PMX}
- Crossover rates: {0.5, 0.6, 0.7, 0.8, 0.9}
- Mutation rates: {0.08, 0.09, 0.1, 0.11, 0.12}.

The three landmarker GAs have a crossover rate of 0.6 and a mutation rate of 0.12, combined with one of the three crossover types, respectively. They are excluded from the set of alternatives to be ranked. The label and dyad rankers are faced with rankings under different conditions $(M, N)$, with $N$ the number of training instances and $M$ the average length of the rankings ($M$ of the 72 alternatives are chosen at random while the others are discarded).

The results in Table 1 are quite consistent with our first study and again confirm that additional information about labels can principally be exploited by a learner to achieve better predictive performances. In particular, BilinPL is

[4] The data set can be obtained from https://www.cs.uni-paderborn.de/fachgebiete/intelligente-systeme/

**Table 1.** Average performance in terms of Kendall's tau and standard deviations of different meta-learners and different conditions (average rankings lengths $M$ and the numbers of training instances $N$).

| M | N | AR | BilinPL | CC | LinPL | LinSidePL | RPC |
|---|---|---|---|---|---|---|---|
| 5 | 30 | 0.192 ± 0.063 | **0.727 ± 0.014** | 0.290 ± 0.063 | 0.317 ± 0.049 | 0.663 ± 0.031 | 0.158 ± 0.052 |
| | 60 | 0.358 ± 0.046 | **0.766 ± 0.014** | 0.428 ± 0.040 | 0.452 ± 0.041 | 0.681 ± 0.026 | 0.311 ± 0.038 |
| | 90 | 0.404 ± 0.030 | **0.770 ± 0.014** | 0.573 ± 0.042 | 0.575 ± 0.037 | 0.691 ± 0.018 | 0.372 ± 0.035 |
| | 120 | 0.430 ± 0.029 | **0.777 ± 0.009** | 0.610 ± 0.031 | 0.619 ± 0.022 | 0.697 ± 0.015 | 0.387 ± 0.032 |
| 10 | 30 | 0.423 ± 0.054 | **0.775 ± 0.007** | 0.539 ± 0.054 | 0.551 ± 0.049 | 0.696 ± 0.018 | 0.397 ± 0.043 |
| | 60 | 0.487 ± 0.017 | **0.781 ± 0.004** | 0.690 ± 0.021 | 0.696 ± 0.013 | 0.718 ± 0.012 | 0.493 ± 0.037 |
| | 90 | 0.523 ± 0.014 | **0.781 ± 0.007** | 0.726 ± 0.015 | 0.726 ± 0.012 | 0.727 ± 0.010 | 0.576 ± 0.018 |
| | 120 | 0.522 ± 0.015 | **0.783 ± 0.006** | 0.750 ± 0.014 | 0.748 ± 0.014 | 0.735 ± 0.011 | 0.620 ± 0.020 |
| 20 | 30 | 0.516 ± 0.037 | **0.781 ± 0.005** | 0.722 ± 0.019 | 0.722 ± 0.015 | 0.728 ± 0.014 | 0.622 ± 0.018 |
| | 60 | 0.549 ± 0.014 | **0.784 ± 0.005** | 0.763 ± 0.013 | 0.758 ± 0.014 | 0.741 ± 0.015 | 0.714 ± 0.022 |
| | 90 | 0.561 ± 0.014 | **0.787 ± 0.006** | 0.779 ± 0.010 | 0.774 ± 0.013 | 0.750 ± 0.015 | 0.751 ± 0.021 |
| | 120 | 0.571 ± 0.022 | **0.787 ± 0.008** | 0.786 ± 0.010 | 0.782 ± 0.010 | 0.758 ± 0.013 | 0.772 ± 0.014 |
| 30 | 30 | 0.554 ± 0.028 | **0.782 ± 0.005** | 0.753 ± 0.013 | 0.746 ± 0.018 | 0.734 ± 0.015 | 0.717 ± 0.019 |
| | 60 | 0.567 ± 0.008 | **0.785 ± 0.003** | 0.782 ± 0.007 | 0.775 ± 0.009 | 0.751 ± 0.010 | 0.767 ± 0.011 |
| | 90 | 0.578 ± 0.008 | 0.787 ± 0.004 | **0.791 ± 0.005** | 0.786 ± 0.005 | 0.758 ± 0.010 | 0.781 ± 0.006 |
| | 120 | 0.580 ± 0.011 | 0.786 ± 0.006 | **0.794 ± 0.005** | 0.789 ± 0.007 | 0.761 ± 0.011 | 0.787 ± 0.005 |

able to take advantage of this information for small values of $M$ and favorably compares to the other label rankers (and, in addition, has of course the advantage of being able to rank GA variants that have not been used in the training phase). As expected, standard label rankers (in this case, CC) surpass BilinPL only for a sufficiently large amount of training data.

## 6 Summary and Outlook

In this paper, we proposed dyad ranking as an extension of the label ranking problem, a specific type of preference learning problem in which preferences on a finite set of choice alternatives are represented in the form of a contextualized ranking. While the context is described in terms of a feature vector, the alternatives are merely identified by their label.

In practice, however, information about properties of the alternatives is often available, too, and such information could obviously be useful from a learning point of view. In dyad ranking, not only the context but also the alternatives are therefore characterized as feature vectors.

The concrete method we developed, BilinPL, is a generalization of an existing label ranking method based on the Plackett-Luce model. First experimental results using synthetic data as well as a case study in meta-learning confirm

that BilinPL tends to be superior to standard label ranking methods if feature information about alternatives is available, at least if training data is scarce in comparison to the number of alternatives to be ranked.

Since the PL approach is only one among several existing label ranking methods, one may wonder to what extent other methods are amenable to the incorporation of label features. This is a question we seek to address in future work. Another interesting idea is to combine label ranking with (unsupervised) representation learning for feature construction [1]: first, labels are embedded in a feature space so as to reflect their similarity in a proper way, and the feature representation thus produced is then used in dyad ranking. Last but not least, there are several interesting applications of dyad ranking, notably those in which standard label ranking has already been used, though without exploiting feature information about choice alternatives. An example of that kind is preference-based reinforcement learning, where label ranking is used to sort actions given states [8]. Since actions do have a natural representation in terms of features or parameters in many reinforcement learning problems, there is obviously scope for enhancement through the incorporation of dyad ranking.

## References


1. Bengio, Y., Courville, A.C., Vincent, P.: Representation learning: A review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1798–1828 (2013)
2. Brazdil, P., Giraud-Carrier, C., Soares, C., Vilalta, R.: Metalearning: Applications to Data Mining, 1st edn. Springer Publishing Company, Incorporated (2008)
3. Brazdil, P., Soares, C., Coasta, J.P.D.: Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. Machine Learning **50**, 251–277 (2003)
4. Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., Li, H.: Learning to rank: from pairwise approach to listwise approach. In: Proceedings of the 24th International Conference on Machine learning (ICML 2007), pp. 129–136. ACM (2007)
5. Cheng, W., Dembczynski, K., Hüllermeier, E.: Label ranking methods based on the Plackett-Luce model. In: Proceedings of the 27th International Conference on Machine Learning (ICML 2010), pp. 215–222 (2010)
6. Cohen, W.W., Schapire, R.E., Singer, Y.: Learning to order things. Journal of Artificial Intelligence Research **10**(1), 243–270 (1999)
7. Fürnkranz, J., Hüllermeier, E.: Preference learning: An introduction. Preference Learning (2010)
8. Fürnkranz, J., Hüllermeier, E., Cheng, W., Park, S.H.: Preference-based reinforcement learning: A formal framework and a policy iteration algorithm. Machine Learning **89**(1), 123–156 (2012)
9. Har-Peled, S., Roth, D., Zimak, D.: Constraint classification: a new approach to multiclass classification. In: Cesa-Bianchi, N., Numao, M., Reischuk, R. (eds.) ALT 2002. LNCS (LNAI), vol. 2533, pp. 365–379. Springer, Heidelberg (2002)
10. Har-Peled, S., Roth, D., Zimak, D.: Constraint classification for multiclass classification and ranking. In: Becker, S., Thrun, S., Obermayer, K. (eds.) Advances in Neural Information Processing Systems, vol. 15, pp. 809–816. MIT Press (2003)

11. Hüllermeier, E., Cheng, W.: Superset learning based on generalized loss minimization. In: Proc. ECML/PKDD 2015, European Conference on Machine Learning and Knowledge Discovery in Databases, Porto, Portugal (2015)
12. Hüllermeier, E., Fürnkranz, J., Cheng, W., Brinker, K.: Label ranking by learning pairwise preferences. Artificial Intelligence **172**(16), 1897–1916 (2008)
13. Kamishima, T., Kazawa, H., Akaho, S.: A survey and empirical comparison of object ranking methods. In: Fürnkranz, J., Hüllermeier, E. (eds.) Preference Learning, pp. 181–201. Springer (2011)
14. Kanda, J., Soares, C., Hruschka, E., de Carvalho, A.: A meta-learning approach to select meta-heuristics for the traveling salesman problem using MLP-based label ranking. In: Huang, T., Zeng, Z., Li, C., Leung, C.S. (eds.) ICONIP 2012, Part III. LNCS, vol. 7665, pp. 488–495. Springer, Heidelberg (2012)
15. Kendall, M.G.: A new measure of rank correlation. Biometrika **30**(1/2), 81–93 (1938)
16. Larranaga, P., Kuijpers, C.M.H., Murga, R.H., Inza, I., Dizdarevic, S.: Genetic algorithms for the traveling salesman problem: A review of representations and operators. Artificial Intelligence Review **13**, 129–170 (1999)
17. Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. Mathematical Programming **45**(1–3), 503–528 (1989)
18. Marden, J.I.: Analyzing and Modeling Rank Data, 1st edn. Chapman & Hall (1995)
19. Menon, A.K., Elkan, C.: Dyadic prediction using a latent feature log-linear model (2010). arXiv preprint arXiv:1006.2156
20. Menon, A.K., Elkan, C.: A log-linear model with latent features for dyadic prediction. In: Proceedings of the 2010 IEEE International Conference on Data Mining, ICDM 2010, pp. 364–373. IEEE Computer Society (2010)
21. Menon, A.K., Elkan, C.: Predicting labels for dyadic data. Data Mining and Knowledge Discovery **21**(2), 327–343 (2010)
22. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press, Cambridge (1998)
23. Pahikkala, T., Airola, A., Stock, M., De Baets, B., Waegeman, W.: Efficient regularized least-squares algorithms for conditional ranking on relational data. Machine Learning **93**, 321–356 (2013)
24. Pahikkala, T., Waegeman, W., Airola, A., Salakoski, T., De Baets, B.: Conditional ranking on relational data. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010, Part II. LNCS, vol. 6322, pp. 499–514. Springer, Heidelberg (2010)
25. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. Journal of Machine Learning Research, 1453–1484 (2005)
26. Vembu, S., Gärtner, T.: Label ranking algorithms: a survey. In: Preference Learning, pp. 45–64. Springer (2011)
27. Zhou, Y., Liu, Y., Yang, J., He, X., Liu, L.: A taxonomy of label ranking algorithms. Journal of Computers **9**(3), 557–565 (2014)