# Plackett-Luce Networks for Dyad Ranking

Dirk Schäfer[1] and Eyke Hüllermeier[2]

[1] University of Marburg, Germany
dirk.schaefer@uni-marburg.de
[2] Department of Computer Science
Paderborn University, Germany
eyke@upb.de

**Abstract.** We propose a new method for dyad ranking, a problem that was recently introduced in the realm of preference learning. Our method, called PLNet, combines a statistical model for rank data, namely the Plackett-Luce model, with neural networks (feed-forward multi-layer perceptrons) in order to learn joint-feature representations for dyads, which are pairs of objects from two domains. The efficacy of PLNet is shown by comparing it experimentally with state-of-the-art methods for dyad and label ranking.

**Keywords:** Preference learning, dyad ranking, Plackett-Luce model, multi-layer perceptrons.

## 1 Introduction

A specific problem in the realm of preference learning [7] is the problem of *label ranking*, which consists of learning a model that maps instances to rankings (total orders) over a finite set of predefined alternatives [20]. 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.

In [17], we introduced *dyad ranking* as a practically motivated generalization of the label ranking problem. In dyad ranking, not only the instances but also the alternatives are represented in terms of attributes—a dyad is a pair consisting of an instance and an alternative. Moreover, for learning in the setting of dyad ranking, we proposed an extension of an existing label ranking method based on the Plackett-Luce model, a statistical model for rank data. This approach is based on modeling latent utility scores of dyads in the form of a Kronecker product of the feature representations of the instance and alternative, respectively, which is why we speak of a *bilinear* Plackett-Luce model.

In this paper, we propose a variant of this approach, called *PLNet*, that allows for modeling latent utilities in a more flexible way. Instead of assuming a

bilinear structure of the joint-feature representation, we model utilities as (feed-forward) neural networks; thanks to the hidden layer of such networks, important non-linear dependencies can thus be captured [16].

The rest of the paper is organized as follows. We provide a formal description of the dyad ranking problem in Section 2 and an overview of related methods in Section 3. Our new method PLNet is described in Section 4. Experimental results are presented in Section 5, prior to concluding in Section 6.

## 2 Problem Setting

Formally, a dyad is a pair of feature vectors $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{Z} = \mathbb{X} \times \mathbb{Y}$, where the feature vectors are from two (not necessarily different) domains $\mathbb{X}$ and $\mathbb{Y}$. A single training observation $\rho_n$ $(1 \leq n \leq N)$ takes the form of a dyad ranking

$$\rho_n : \boldsymbol{z}^{(1)} \succ \boldsymbol{z}^{(2)} \succ \ldots \succ \boldsymbol{z}^{(M_n)}, \;\; M_n \geq 2, \tag{1}$$

the length $M_n$ of which can vary between observations in the data set $\mathcal{D} = \{\rho_n\}_{n=1}^{N}$. An equivalent notation for a single training example that will be used later on is a set of dyads

$$\varrho_n = \left\{ \boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \ldots, \boldsymbol{z}^{(M_n)} \right\}$$

together with a permutation $\pi_n : \{1, \ldots, M_n\} \longrightarrow \{1, \ldots, M_n\}$ indicating how these dyads are ranked.

The task of a dyad ranking method is to learn a ranking function that accepts as input any set of (new) dyads and produces as output a ranking of these dyads.

An important special case, called *contextual dyad ranking*, is closely related to label ranking [17]. As already mentioned, the label ranking problem is about learning a model that maps instances to rankings over a finite set of predefined choice alternatives $\mathcal{Y} = \{y_1, \ldots, y_K\}$. In terms of dyad ranking this means that all dyads in an observation 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, (1) can also be written as

$$\rho_n : \left(\boldsymbol{x}, \boldsymbol{y}^{(1)}\right) \succ \left(\boldsymbol{x}, \boldsymbol{y}^{(2)}\right) \succ \ldots \succ \left(\boldsymbol{x}, \boldsymbol{y}^{(M_n)}\right) \;\; . \tag{2}$$

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 $\boldsymbol{x}$. Contextual dyad ranking generalizes label ranking by considering feature vectors instead of labels. This includes vector space embeddings of labels and additional descriptions (side-information) about labels. Contextual dyadic preferences are equivalent to training triplets of the form $y^{(1)} \succ_{\boldsymbol{x}} y^{(2)}$, which are encountered frequently in applications such as similarity learning.

## 3 Related Work

The Bilinear Plackett-Luce model (BilinPL) for dyad ranking was introduced in [17]. It builds on a statistical ranking model introduced by Plackett and Luce [14] and represents the parameters of this model as a log-bilinear function $f(\boldsymbol{x}, \boldsymbol{y}) = \exp g(\boldsymbol{x}, \boldsymbol{y})$, which takes as input the dyad member feature vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. The output of the function is a real positive value, which can be interpreted as a utility score. The bilinear function $g(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^\top \boldsymbol{W} \boldsymbol{y}$ can equivalently be expressed as the function $g(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{w}, \boldsymbol{x} \otimes \boldsymbol{y} \rangle$, i.e., the dot product between a weight vector and a joint-feature vector consisting of cross-products (also known as the Kronecker product). This kind of joint-feature vector formulation strongly depends on the dyad feature vectors, and the representation bias it imposes on the model can be suboptimal.

Comparison training refers to a framework introduced for learning from pairwise preferences with a neural network [19]. The network architecture consists of two subnetworks which are connected to a single output node and indicates which of the two inputs is preferred over the other. The weights associated with the last hidden layer of one subnetwork is the mirrored version of the other subnetwork's weights. This setup solves two principal problems, namely efficiency and consistency. In the evaluation phase it is sufficient to use just one subnetwork and evaluate $n$ alternatives instead of $\mathcal{O}(n^2)$ pairs. Furthermore, this kind of network architecture also enforces transitivity of the predicted preferences. Although many modifications of the original comparison training have been proposed in the past (see e.g. the survey [6]), its inputs however are essentially restricted to pairwise training signals.

The aforementioned label ranking problem has been tackled with neural network approaches previously [15, 11]. A multi-layer perceptron (MLP) has been utilized to produce recommendations on meta-heuristics (labels) on different meta-features (instances) within the realm of meta-learning [11]. This kind of neural network exhibits an output layer with as many nodes as there are labels. The error signal used to modify the network's weights is formed by using the mean squared error on the target rank positions of the labels. In [15] more effort has been spent to incorporate label ranking loss information into the backpropagation procedure. To this end some variations for this procedure have been investigated. Both architectures are similar to each other and have two essential limitations: first, they depend on a fixed number of labels, and second, they cannot cope with incomplete ranking observations. In addition, they lack the ability to provide probabilistic information on their predictions.

In the domain of information retrieval, the neural network-based approaches RankNet and ListNet have a probabilistic foundation [2, 3]. RankNet [2] uses pairwise inputs to learn a utility scoring function with the cross entropy loss. To this end, the training data consists of sample pairs together with target probabilities. These quantify the probability to rank the first sample higher than the second. With the introduction of target probabilities, this approach enables the possibility of modeling ties between samples. ListNet [3, 13] similarly uses the cross entropy as a metric, but in contrast to RankNet it processes lists of

samples instead of pairwise preferences as basic observation. There are, however, some important differences between ListNet and our approach:

- The learning approach in ListNet addresses only a special case of the Plackett-Luce distribution, namely the case of Top-k data with $k = 1$.
- In ListNet, a *linear* neural network is used. This is in contrast to our approach, in which non-linear relationships between inputs and outputs are learned. Linearity in the ListNet approach implies that much emphasize must be put on engineering joint feature input vectors.
- In ListNet, the query-document features are associated with absolute scores (relevance degrees) as training information, i.e., quantitative data, whereas PLNet deals with rankings, i.e., data of qualitative nature.[3]

ListNet as well as RankNet expect single instance joint-feature vectors $\boldsymbol{x} = \Psi(q, d)$ as inputs that are created from query-document pairs $(q, d)$ via feature-engineering. These approaches are closely related to our method, though with the major difference of ranked dyads instead of absolute ratings as training input; and perhaps more importantly, our goal is to *learn* joint-feature representations instead of engineering them.

## 4 Plackett-Luce Network

Throughout this section we use the following notation. A ranking of dyads is represented in terms of a permutation $\pi$. More specifically, given a numbering of dyads from 1 to $M$, let $\pi(i)$ be the number $j$ of the dyad put on the $i$-th position in the ranking; the inverse $\pi^{-1}(j)$ denotes the rank position of the dyad specified by index $j$.

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

$$\mathbf{P}(\pi \mid \boldsymbol{v}) = \prod_{i=1}^{K} \frac{v_{\pi(i)}}{v_{\pi(i)} + v_{\pi(i+1)} + \ldots + v_{\pi(K)}} = \prod_{i=1}^{K-1} \frac{v_{\pi(i)}}{\sum_{j=i}^{K} v_{\pi(j)}} \tag{3}$$

Obviously, the Plackett-Luce model is only determined up to a positive multiplicative constant, i.e., $\mathbf{P}(\pi \mid \boldsymbol{v}) \equiv \mathbf{P}(\pi \mid s \cdot \boldsymbol{v})$ for all $s > 0$.

As an appealing property of the PL model, we also note that its marginal probabilities (probabilities of rankings on subsets of alternatives) are easy to compute and can be expressed in closed form. More specifically, the marginal of a PL model on $M < K$ alternatives $y_{i(1)}, \ldots, y_{i(M)}$ is again a PL model with parameters $v_{i(1)}, \ldots, v_{i(M)}$.

[3] The use of query-document-associated scores as PL model parameters is arguable.

### 4.2 Architecture

The core idea of the Placket-Luce Network (PLNet) is to learn (latent) utility functions $u = g(\boldsymbol{x}, \boldsymbol{y})$, where $g$ is a single multi-layer feed-forward neural network. Since we are interested in utility scores that reflect rankings, we utilize the Plackett-Luce model and express its real-valued skill parameters as functions of the form $\log(v) = u = g(\boldsymbol{x}, \boldsymbol{y})$. Thus, probabilities of rankings are given by

$$P(\pi \mid \varrho, \boldsymbol{u}) = \prod_{k=1}^{K-1} \frac{\exp(u_{\pi(k)})}{\sum_{l=k}^{K} \exp(u_{\pi(l)})} \ . \tag{4}$$

The PLNet is a multi-layer perceptron (MLP), which is constructed as shown in Figure 1. It consists of multiple layers and takes as input two vectors corresponding to the members of a dyad. There is at least one hidden layer with nodes using a sigmoidal activation function to learn non-linear mappings. The output layer produces a scalar value of the form $u = \langle \boldsymbol{w}^L, \boldsymbol{a}^{L-1} \rangle + b^L$. Technically, a bias term $b^L$ is actually not needed, as it has no effect on the PLNet model (4): With the choice of the exponential function (to ensure positivity of the parameters $v$) and the independence of $b^L$ of the inputs $\boldsymbol{x}$ and $\boldsymbol{y}$, we have

$$v = \exp\left(\langle w^L, a^{L-1} \rangle + b^L\right) = b^L \cdot \exp\left(\langle w^L, a^{L-1} \rangle\right) \ ,$$

and as noted before, the PL model is invariant toward multiplication of the weights with a positive scalar.

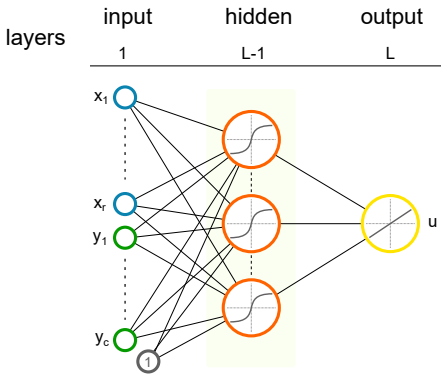


**Fig. 1.** PLNet architecture. This kind of feed-forward neural network is composed of several layers. Dyad members $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{y})$ are entered at the input layer, whose nodes are fully connected with the nodes of the next layer. Inner (hidden) layers have nodes endowed with a non-linear activation function. The output layer consists of a single node with linear activation function.

### 4.3 Training

The training procedure uses *back-propagation*, which is a gradient technique to find optimal weights [16]. Following [1], a feed-forward network can be seen as a framework for modeling the conditional probability distribution. For a set of training data $\{\varrho_n, \pi_n\}$ with

$$\varrho_n = \left\{ \left(\boldsymbol{x}_n^{(1)}, \boldsymbol{y}_n^{(1)}\right), \ldots, \left(\boldsymbol{x}_n^{(M_n)}, \boldsymbol{y}_n^{(M_n)}\right) \right\} \ ,$$

the likelihood can be written as

$$\mathcal{L} = \prod_n \mathbf{P}(\varrho_n, \pi_n) = \prod_n \mathbf{P}(\pi_n \mid \varrho_n)\mathbf{P}(\varrho_n)$$

if we assume the observations $\{\varrho_n, \pi_n\}$ to be independent and identically distributed. It is generally more convenient to minimize the logarithm of the likelihood, hence we aim to minimize the following error function:

$$E = -\ln \mathcal{L} = -\sum_n \ln \mathbf{P}(\pi_n \mid \varrho_n) - \sum_n \mathbf{P}(\varrho_n) \tag{5}$$

The second term in (5) can be dropped, as it does not depend on the network parameters (and only represents an additive constant). More specifically, we denote the negative log-likelihood (NLL) by $E = \sum_n E_n$, where

$$E_n = -\log P(\pi_n \mid \varrho_n, \boldsymbol{u}) = \sum_{k=1}^{M_n - 1} \log \sum_{l=k}^{M_n} \exp\left(u_{\pi(l)}\right) - \sum_{k=1}^{M_n - 1} u_{\pi(k)} \ . \tag{6}$$

The errors propagated back to the network can thus be expressed as the (partial) derivatives of the NLL:

$$\delta_i^L \equiv \frac{\partial E_n}{\partial u_i} = \sum_{k=1}^{M_n - 1} \frac{\mathbb{1}_{\{\pi^{-1}(i) \geq k\}} \exp(u_i)}{\sum_{l=k}^{M_n} \exp(u_{\pi(l)})} - \mathbb{1}_{\{i \leq M_n - 1\}} \tag{7}$$

Note that the calculation of the error in (7) depends on all the $M_n$ utilities of a training sample. To this end, we propose a training procedure as depicted in Figure 2, where we run $M_n$ feed-forward calculations for a sample (consisting of $M_n$ dyads) in parallel to obtain utility values. The $\delta$-values obtained according to (7) can then be used for the standard back-propagation procedure on each of the parallel $M_n$ copies of the master network. After performing the weight updates individually, the master's weights can be updated by aggregating the individual weight changes. In the implementation we applied the following simple strategy: $w_{ji} = w_{ji} - \eta \cdot \Delta w_{ji}^k$, $1 \leq k \leq M_n$, where $\eta$ denotes the learning rate. The procedure is then repeated several times with the training samples and stopped when the error has diminished sufficiently.

Another point to be mentioned is regularization. Here, we suggest to use *early stopping* by tracking the NLL values of the training and validation data during the learning process. A good point to stop the training and to prevent over-fitting is when the validation NLL values begin to rise again.

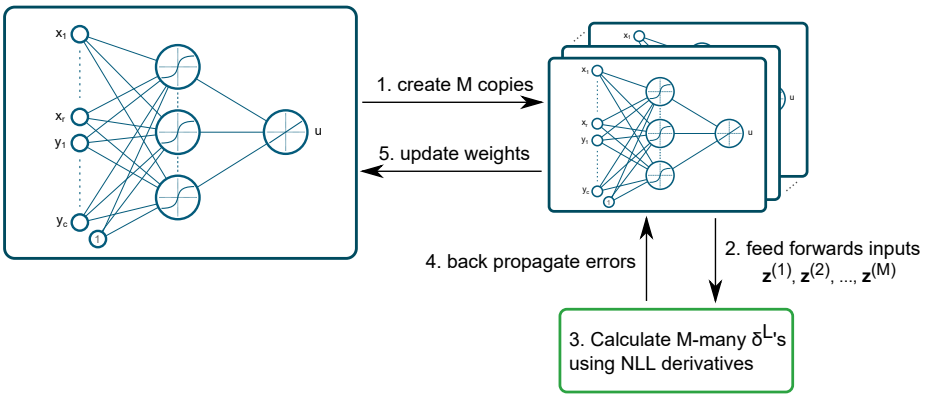


**Fig. 2.** PLNet training steps. An observation consists of $M$ ranked dyads, which are fed into $M$ copies of the current PLNet model. The output of this composite feed-forward procedure (step 2) is used to calculate the ranking cost (step 3). Finally, the weights are adjusted via back-propagation (step 4), and the changes in the weights are merged into a single network (step 5).

### 4.4 PLNet for Label Ranking

In order to apply PLNet on the problem of label ranking, one needs a vector representation for labels. Unless other label features are available, a natural approach is to a use 1-of-K encoding. Thus, given $K$ labels, the $i$-th label is represented by the vector $\boldsymbol{e}_i \in \{0,1\}^K$ with entry 1 at position $i$ and 0 otherwise.

## 5 Experiments

In the following experiments, the predictive performance is measured in terms of the Kendall's tau coefficient [12], a rank correlation measure commonly used in the statistical and preference learning literature [20, 21]. It is defined as

$$\tau = \frac{C(\pi, \hat{\pi}) - D(\pi, \hat{\pi})}{K(K-1)/2} \quad , \tag{8}$$

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 alternatives). 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

Here, we compare the dyad ranking methods PLNet and BilinPL. To this end, we sample from a PLNet with three layers. Its inputs are dyads composed of

one-dimensional vectors, i.e., $\boldsymbol{x}, \boldsymbol{y} \in [-1, 1]$. The hidden layer has 25 nodes and all weights are initialized at random by sampling from the uniform distribution in $[-15, 15]$. From a total of 400 dyads, we sample 500 training and 50 test rankings consisting of 5 dyads each.

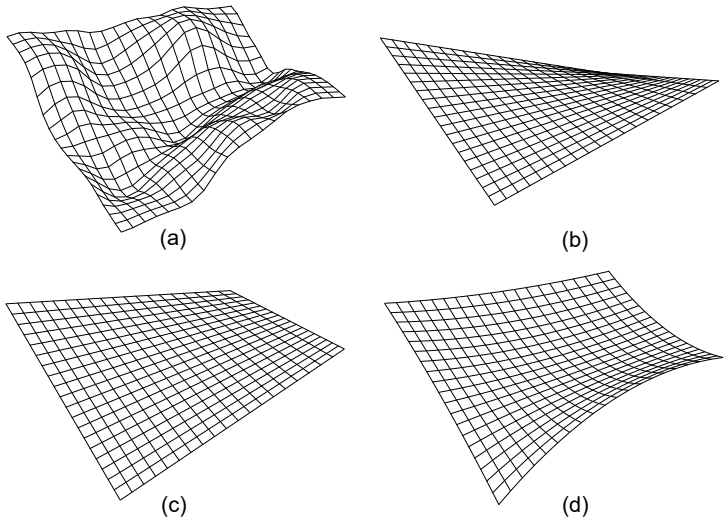


**Fig. 3.** Log-skill landscapes of the methods PLNet and BilinPL. (a) is produced by PLNet after learning from object pair rankings. (b)-(d) are produced by BilinPL using different feature specifications.

For the BilinPL, we choose from three different variants of input features. They are based on the Kronecker product between object pair features to from joint-feature vectors (resulting in first and second order models):

| $\boldsymbol{x}_f$ | $\boldsymbol{y}_f$ | $\Rightarrow \boldsymbol{z}_f$ | Identifier |
|---|---|---|---|
| $x$ | $y$ | $[x \cdot y]$ | BilinPL-1 |
| $[x, 1]$ | $[y, 1]$ | $[x, y, xy, 1]$ | BilinPL-2 |
| $[x, x^2, 1]$ | $[y, y^2, 1]$ | $[x, x^2, y, y^2, xy, xy^2, x^2y, x^2y^2, 1]$ | BilinPL-3 |

Table 1 and the corresponding Figure 3 underpin two key aspects. Firstly, the expressiveness of PLNet can be much larger compared to the BilinPL versions. Secondly, the predictive quality varies strongly for BilinPL depending on the choice of the dyad features.

**Table 1.** Results of the synthetic pair ranking data experiment.

| | PLNet | BilinPL-1 | BilinPL-2 | BilinPL-3 |
|---|---|---|---|---|
| Kendall's $\tau$ | 0.944 | 0.380 | 0.644 | 0.852 |
| Figure 3 | (a) | (b) | (c) | (d) |

### 5.2 UCI Label Ranking Datasets

A suite of benchmark data sets have been established for the label ranking setting [4].[4] They are based on 16 well known multi-class and regression data sets from the UCI repository and processed in two ways to address the ranking problem (see Table 2 for their properties). For data sets of type A (multi-class problems), rankings were generated by training a naive Bayes classifier on the complete data set and ordering the class labels according to the predicted class probabilities for each example. For data sets of type B (regression problems), a subset of instance attributes are removed from the data sets and were interpreted as labels. Rankings were then obtained by standardizing the attributes and then ordering them by size. This approach is justified by assuming that the original attributes are correlated and the remaining features contain information about the rankings of the removed attributes.

We compare the performance of PLNet to other state-of-the-art label rankings methods on these data sets using 10-fold cross-validation. For PLNet, we use three layers with 10 neurons for the hidden layer. In addition to BilinPL, we include Ranking by Pairwise Comparison (RPC, [10]), Constrained Classification (CC, [8, 9]), and the log-linear model for label ranking (LL, [5]) as additional baselines.[5] For BilinPL, we chose as dyad features $x_f = [x, 1]$ and $y_f^{(i)} = e_i$, which by means of the Kronecker product results in a joint-feature vector representation called multi-vector [18].

The results (see Table 3) indicate that PLNet is competitive to the other approaches and most of the time even superior. The performances on data sets where PLNet is less powerful also show its weakness. For data sets consisting of only a few instances but many attributes, PLNet is likely to over-fit. Of course, another issue is the choice of the architecture itself (how many layers, how many nodes, ...). Linear models are advantageous in comparison to PLNet if their inductive bias is correct, which is the case, for example, for the *fried* problem.

## 6 Conclusion

We introduced a new method for the problem of dyad ranking, called PLNet. The method exhibits some interesting properties, notably the following: it is

[4] Available online at https://www-old.cs.uni-paderborn.de/fachgebiete/intelligente-systeme/software/label-ranking-datasets.html.

[5] CC was used in its online variant as described in [10].

**Table 2.** Semi-synthetic label ranking data sets and their properties.

| Type A | | | | Type B | | | |
|---|---|---|---|---|---|---|---|
| data set | # inst.(N) | # attr.(d) | # labels(M) | data set | # inst.(N) | # attr.(d) | # labels(M) |
| authorship | 841 | 70 | 4 | bodyfat | 252 | 7 | 7 |
| glass | 214 | 9 | 6 | calhousing | 20640 | 4 | 4 |
| iris | 150 | 4 | 3 | cpu-small | 8192 | 6 | 5 |
| pendigits | 10992 | 16 | 10 | elevators | 16599 | 9 | 9 |
| segment | 2310 | 18 | 7 | fried | 40769 | 9 | 5 |
| vehicle | 846 | 18 | 4 | housing | 506 | 6 | 6 |
| vowel | 528 | 10 | 11 | stock | 950 | 5 | 5 |
| wine | 178 | 13 | 3 | wisconsin | 194 | 16 | 16 |

**Table 3.** Results on the UCI label ranking data sets.

| data set | BilinPL | CC | LL | PLNet | RPC-LR |
|---|---|---|---|---|---|
| authorship | 0.931±0.013 | 0.916±0.015 | **0.935±0.013** | 0.908±0.025 | 0.917±0.020 |
| bodyfat | 0.268±0.059 | 0.245±0.052 | **0.287±0.060** | 0.251±0.040 | 0.285±0.061 |
| calhousing | 0.220±0.011 | 0.254±0.009 | 0.235±0.010 | **0.272±0.014** | 0.243±0.010 |
| cpu-small | 0.445±0.016 | 0.468±0.017 | 0.431±0.017 | **0.500±0.019** | 0.449±0.016 |
| elevators | 0.730±0.007 | 0.770±0.009 | 0.725±0.006 | **0.788±0.009** | 0.749±0.008 |
| fried | 0.999±0.000 | 0.999±0.000 | 0.997±0.001 | 0.951±0.010 | **1.000±0.000** |
| glass | 0.835±0.072 | 0.830±0.079 | 0.825±0.074 | 0.846±0.080 | **0.889±0.057** |
| housing | 0.655±0.040 | 0.639±0.044 | 0.646±0.034 | **0.703±0.033** | 0.672±0.041 |
| iris | 0.813±0.112 | 0.800±0.109 | 0.804±0.101 | **0.960±0.049** | 0.911±0.047 |
| pendigits | 0.892±0.003 | 0.896±0.002 | 0.879±0.002 | 0.905±0.005 | **0.932±0.002** |
| segment | 0.903±0.008 | 0.910±0.008 | 0.879±0.009 | **0.939±0.008** | 0.929±0.009 |
| stock | 0.704±0.016 | 0.714±0.016 | 0.702±0.018 | **0.882±0.020** | 0.774±0.024 |
| vehicle | 0.855±0.020 | 0.850±0.025 | 0.790±0.023 | **0.872±0.025** | 0.855±0.015 |
| vowel | 0.581±0.026 | 0.577±0.046 | 0.608±0.023 | **0.805±0.016** | 0.644±0.021 |
| wine | 0.929±0.052 | 0.914±0.069 | **0.954±0.041** | 0.942±0.034 | 0.925±0.054 |
| wisconsin | 0.629±0.028 | 0.612±0.030 | 0.605±0.027 | 0.514±0.028 | **0.632±0.027** |
| average rank | 3.250 | 3.625 | 3.812 | 2.125 | 2.188 |

probabilistic in nature, can handle incomplete rankings, and builds on standard neural network components. Thus, it is possible to improve the method further with techniques developed in the neural networks literature during the last years,

especially with recent advances in deep learning. This point is an important aspect of future research. The study of other neural network architectures such as (restricted) Boltzman machines for dyad ranking are also conceivable.

## References


1. Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995.
2. Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings ICML, 22nd International Conference on Machine Learning*, pages 89–96, New York, NY, USA, 2005. ACM.
3. Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: From pairwise approach to listwise approach. In *Proceedings ICML, 24th International Conference on Machine Learning*, pages 129–136, New York, NY, USA, 2007. ACM.
4. Weiwei Cheng, Jens Hühn, and Eyke Hüllermeier. Decision tree and instance-based learning for label ranking. In *Proceedings ICML, 26th International Conference on Machine Learning*, pages 161–168, Montreal, Canada, June 2009. Omnipress.
5. Ofer Dekel, Yoram Singer, and Christopher D Manning. Log-linear models for label ranking. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16 (NIPS-2003)*, pages 497–504. MIT Press, 2003.
6. Johannes Fürnkranz. Machine learning in games: A survey. In Johannes Fürnkranz and M. Kubat, editors, *Machines that Learn to Play Games*, chapter 2, pages 11–59. Nova Science Publishers, Huntington, NY, 2001.
7. Johannes Fürnkranz and Eyke Hüllermeier. *Preference Learning*. Springer, New York, NY, USA, 2011.
8. Sariel Har-Peled, Dan Roth, and Dav Zimak. Constraint classification: A new approach to multiclass classification. In *Proceedings ALT, 13th International Conference on Algorithmic Learning Theory*.
9. Sariel Har-Peled, Dan Roth, and Dav Zimak. Constraint classification for multiclass classification and ranking. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 809–816. MIT Press, 2002.
10. Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16):1897–1916, 2008.
11. Jorge Kanda, Carlos Soares, Eduardo R. Hruschka, and André Carlos Ponce Leon Ferreira de Carvalho. A meta-learning approach to select meta-heuristics for the traveling salesman problem using MLP-based label ranking. In *Proceedings ICONIP, 19th International Conference on Neural Information Processing*, pages 488–495, Doha, Qatar, 2012. Springer.
12. M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.
13. Tianyi Luo, Dong Wang, Rong Liu, and Yiqiao Pan. Stochastic top-k listnet. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 676–684, Lisbon, Portugal, September 2015. Association for Computational Linguistics.
14. John I. Marden. *Analyzing and Modeling Rank Data*. Chapman & Hall, 1995.

15. Geraldina Ribeiro, Wouter Duivesteijn, Carlos Soares, and Arno J. Knobbe. Multilayer perceptron for label ranking. In *Proceedings ICANN, 22nd International Conference on Artificial Neural Networks*, pages 25–32, Lausanne, Switzerland, 2012. Springer.
16. David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:9, 1986.
17. Dirk Schäfer and Eyke Hüllermeier. Dyad ranking using a bilinear Plackett-Luce model. In *Proceedings ECML/PKDD–2015, European Conference on Machine Learning and Knowledge Discovery in Databases*, Porto, Portugal. Springer.
18. Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, New York, NY, USA, 2014.
19. Gerald Tesauro. Connectionist learning of expert preferences by comparison training. In D. Touretzky, editor, *Advances in Neural Information Processing Systems 1 (NIPS-1988)*, pages 99–106. Morgan Kaufmann, 1989.
20. Shankar Vembu and Thomas Gärtner. Label ranking algorithms: A survey. In J. Fürnkranz and E. Hüllermeier, editors, *Preference Learning*, pages 45–64. Springer-Verlag, 2011.
21. Yangming Zhou, Yangguang Liu, Jiangang Yang, Xiaoqi He, and Liangliang Liu. A taxonomy of label ranking algorithms. *Journal of Computers*, 9(3):557–565, 2014.