# Multilayer Perceptron for Label Ranking

Geraldina Ribeiro[1], Wouter Duivesteijn[2], Carlos Soares[3], and Arno Knobbe[2]

[1] Faculdade de Economia, Universidade do Porto, Portugal
100414012@fep.up.pt
[2] LIACS, Leiden University, the Netherlands
{wouterd,knobbe}@liacs.nl
[3] INESC TEC, Universidade do Porto, Portugal
csoares@fep.up.pt

**Abstract.** Label Ranking problems are receiving increasing attention in machine learning. The goal is to predict not just a single value from a finite set of labels, but rather the permutation of that set that applies to a new example (e.g., the ranking of a set of financial analysts in terms of the quality of their recommendations). In this paper, we adapt a multilayer perceptron algorithm for label ranking. We focus on the adaptation of the Back-Propagation (BP) mechanism. Six approaches are proposed to estimate the error signal that is propagated by BP. The methods are discussed and empirically evaluated on a set of benchmark problems.

**Keywords:** Label ranking, back-propagation, multilayer perceptron.

## 1 Introduction

In many real-world applications, assigning a single label to an example is not enough. For instance, when trading in the stock market based on recommendations from financial analysts, predicting who is the best analyst does not suffice because 1) he/she may not make a recommendation in the near future and 2) we may prefer to take into account recommendations of multiple analysts, to be on the safe side [1]. Hence, to support this approach, a model should predict a ranking of analysts rather than suggesting a single one. Such a situation can be modeled as a Label Ranking (LR) problem: a form of preference learning, aiming to predict a mapping from examples to rankings of a finite set of labels [2].

Recently, quite some solutions have been proposed for the label ranking problem [2], including one based on the Multilayer Perceptron algorithm (MLP) [4]. MLP is a type of neural network architecture, which has been applied in a supervised learning context using the error back-propagation (BP) learning algorithm. In this paper, we try a different approach to the simple adaptation proposed earlier [4]. We adapt the BP learning mechanism to LR. More specifically, we investigate how the error signal explored by BP can use information from the LR loss function. We introduce six approaches and evaluate their (relative) performance. We also show some preliminary experimental results that indicate whether our new method could compete with state-of-the-art LR methods.

The remainder of this paper is organized as follows. Section 2 formalizes the LR problem and recalls the BP algorithm for neural networks. In Section 3 we introduce our new adaptation of a multilayer perceptron to solve the LR problem, and the approaches created to estimate the error signal. The experimental results are presented in Section 4, and Section 5 concludes this paper.

## 2 Preliminaries

Throughout this paper, we assume a training set $\mathcal{T} = \{\langle x_n, \pi_n \rangle\}$ consisting of $t$ examples $x_n$ and their associated label rankings $\pi_n$. Such a ranking is a permutation of a finite set of labels $\mathcal{L} = \{\lambda_1, \dots, \lambda_k\}$, given $k$ , taken from the permutation space $\Omega_{\mathcal{L}}$. Each example $x_n$ consists of $m$ attributes $x_n = \{a_1, \dots, a_m\}$ and is taken from the example space $X$. The position of $\lambda_a$ in a ranking $\pi_n$ is denoted by $\pi_n(a)$ and assumes a value in the set $\{1, \dots, k\}$.

### 2.1 Label Ranking

Given $\mathcal{T} = \{\langle x_n, \pi_n \rangle\}$, the goal in LR is to learn a function $f : X \rightarrow \Omega_{\mathcal{L}}$ that minimizes a given loss function function $l = \frac{1}{t} \sum_{n=1}^{t} \tau(\pi_n, \hat{\pi_n})$. With this mapping, we are able to predict a ranking $\hat{\pi_n}$ of the labels in $\mathcal{L}$ for a new example $x_n$. Loss functions in LR are typically based on measures of rank correlation, that assess the similarity between two rankings. One such measure is Kendall's $\tau$ coefficient, denoted $\tau(\hat{\pi_n}, \pi_n)$. The LR error is defined as $e_\tau(n)$ on the $n^{\text{th}}$ training example by $e_\tau(n) = 1/2 - 1/2 \cdot \tau(\hat{\pi_n}, \pi_n)$. The LR error always lies between 0 and 1, where $e_\tau(n) = 0$ means that the network returns a prediction equal to $\pi_n$ and $e_\tau(n) = 1$ means that the labels in $\hat{\pi_n}$ are sorted in the reverse order of $\pi_n$.

There are different approaches to solve LR problems. In reduction techniques, the method is to learn a utility function for each label using the constraint classification technique [5] or a log-linear model [6]. Ranking by pairwise comparisons [2,7] is a well-known method to model rankings as pairwise binary preferences [5]. In probabilistic discriminative methods, the purpose is to estimate a distribution for the probability of a ranking given an example [2,8,9]. Another approach is adapting a machine learning algorithm based on similarity between rankings. In [10], an adaptation of association rules was created where the goal is to discover frequent pairs of attributes associated with a ranking. In [1], an adaptation of a naive Bayes model is proposed where probabilities are replaced by the concept of distance between rankings. In [4], different architectures of an MLP are used to obtain a ranking prediction. In this paper, we propose an adaptation of an MLP for LR problems based on similarity measures.

### 2.2 Neural Networks and Back-Propagation

An MLP is a particular form of a Neural Network (NN), a computational model often used to solve learning problems [11]. It consists of a weighted directed graph of an interconnected set of neurons organized in separate layers: the input layer, the hidden layer(s) and the output layer. Each layer has one or more neurons. Every neuron $i$ is connected to the $j$ neurons of the next layer by a set of weighted links denoted by $w_{1i}, \ldots, w_{ji}$. At the input layer, $\{a_1, a_2, \ldots, a_m\}$ represent $m$ input signals associated with the $m$ attributes. At the hidden and output layers, each neuron $j$ receives the input signals as a linear combination of the output given by: $v_j = \sum_{i=0}^{m} w_{ji} a_i$. The linear combinations are transformed into output signals using an activation function $\varphi(v_j)$. These signals are sent in a forward direction layer by layer to the output layer which delivers an output $y_j$ for each output neuron $j$. In classification, each class is associated with an output neuron and the prediction is typically given by the one with the highest activation level.

The goal is to define the values for the connections weights that return the outputs which lowest error, i.e., the output is most similar to the desired value, $d(n)$. One method to learn the weights is BP, which propagates errors in a backward direction from the output layer to the input layer, updating the weight connections if an error is detected at the output layer. A weight correction on the $n^{\text{th}}$ training example is defined in terms of the error signals $c_j(n)$ for each output neuron $j$. Considering a sequential mode in which the weights are updated after every training example, the predicted output $y_j(n)$ is compared with the desired target $d_j(n)$, and the individual error $e_j(n)$ is estimated as follows: $e_j(n) = d_j(n) - y_j(n)$. In a typical NN, the error signal is equal to the individual error, because the predicted output is directly compared with the target. The correction is given by $\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n)$, where $\eta$ is the learning rate, $y_i(n)$ is the output signal of the previous neuron $i$ and the local gradient $\delta_j$ is defined by $\delta_j = e_j(n) \varphi'(v_j(n))$. For a hidden neuron $i$, the local gradient is defined in a recursive form by $\delta_i(n) = \varphi_i'(v_i(n)) \sum_j \delta_j(n) w_{ji}(n)$.

To prevent the MLP learning from getting stuck in a local optimum we use random-restart hill climbing, by generating new random weights $w_{ji} \sim \mathcal{N}(0, 1)$. For each restart we present every example in the training set to the learning process a user-defined number of times, called an *epoch*. The weights associated with the best performance are returned.

## 3 Multilayer Perceptron for Label Ranking

Our adaptation of MLP for LR essentially consists of 1) the method to generate a ranking from the output layer and 2) the error functions guiding the BP learning process. The output layer contains $k$ neurons (one for each label). The output $y_j$ of a neuron $j$ at the output layer does not represent a target value or class but rather the score associated with a label $\lambda_j$. By ordering all the scores, the predicted ranks $\hat{\pi_n}(j)$ of the label $\lambda_j$ and, thus, the predicted ranking.

The tricky point of adapting an MLP for LR is the weight corrections in the BP process: minimizing the individual errors does not necessarily lead to minimizing the LR loss. We propose six approaches to define the error signal $c_j$ at the output layer. The weight connection $w_{ji}(n)$ is updated based on the estimated $c_j(n)$ using the delta rule $\Delta w_{ji}(n) = \eta c_j(n) y_i(n)$.

*Local Approach (LA).* The error signal is the individual error of each output neuron, $c_j(n) = e_j(n) = \pi_n(j) - \hat{\pi_n}(j)$, as in the original MLP. The LR error, $e_\tau$, is only used to evaluate the activation of the BP.

*Global Approach (GA).* The error signal is defined in terms of the LR error. In this case, it is simply given by $c_j(n) = e_\tau(n)$.

*Combined Approach (CA).* $CA$ is a combination between $GA$ and $LA$, $c_j(n) = e_j(n)e_\tau(n)$. We note that a neuron which returns the correct position $\pi_n(j) = \hat{\pi_n(j)}$ (i.e., $e_j(n) = 0$) is not penalized even if $e_\tau > 0$.

*Weight-Based Signed Global Approach (WSGA).* The error signal is defined in terms of the LR error and the incoming weight connections of the output layer. We assume that a high LR error means that some weights of neurons are too high and other are too low. The output neurons are ranked according to their average weights $\bar{w_j} = \frac{1}{q}\sum_{i=0}^{q} w_{ji}$, resulting in a position $p_w(j) \in [1, \ldots, k]$. The error of the neurons with a position above the mean is negative and it is positive otherwise:

$$c_j(n) = \begin{cases} -e_\tau(n), & \text{if} \quad p_w(j) > \left(\frac{k}{2} + 0.5\right), \\ e_\tau(n), & \text{if} \quad p_w(j) < \left(\frac{k}{2} + 0.5\right), \\ 0, & \text{if} \quad p_w(j) = \left(\frac{k}{2} + 0.5\right). \end{cases} \tag{1}$$

*Score-Based Signed Global Approach (SSGA).* The motivation for SSGA is the same as for WSGA. The difference is that we rank the output neuron scores $y_j$ instead of the input weights. The positions of the weights, $p_w(j)$ is replaced in eq. 1 with the positions of the scores, $p_s(j)$.

*Individual Weight-Based Signed Global Approach (IWSGA).* This assumes that all the weight connections at the output layer are important to define the error signal and are considered independently of the neurons they connect to. The error signal denoted $c_{ji}(n)$ is associated with the weight of the connection between output neuron $i$ and hidden neuron $j$. This is similar to WSGA but we rank all weight connections individually, rather than the average weights for each output neuron. The weight corrections are given by $\Delta w_{ji}(n) = \eta c_{ji}(n) y_i(n)$, where:

$$c_{ji}(n) = \begin{cases} -e_\tau(n), & \text{if} \quad p_{gw}(ji) > \frac{qk}{2}, \\ e_\tau(n), & \text{if} \quad p_{gw}(ji) \leq \frac{qk}{2}. \end{cases}$$

## 4 Experimental Results

The goal is to compare the performance of the proposed approaches on different datasets. The datasets used for the evaluation are from the KEBI Data Repository [12] hosted by the Philipps University of Marburg. These datasets, which are commonly used for LR, are presented in Table 1. Our approach starts

**Table 1.** Datasets for LR

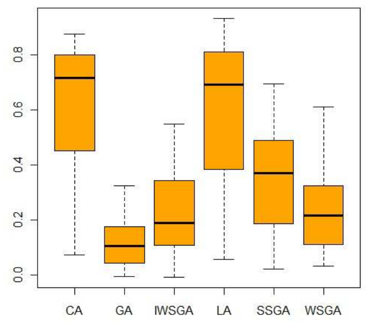
| Dataset | Type | $k$ | $m$ | Instances | Dataset | Type | $k$ | $m$ | Instances |
|---|---|---|---|---|---|---|---|---|---|
| Authorship | A | 4 | **70** | 841 | Iris | A | 3 | 4 | 150 |
| Bodyfat | B | 7 | 7 | 252 | Pendigits | A | 10 | **16** | 10992 |
| Calhousing | B | 4 | 4 | 20640 | Segment | A | 7 | **18** | 2310 |
| Cpu-small | B | 5 | 6 | 8192 | Stock | B | 5 | 5 | 950 |
| Elevators | B | 9 | 9 | 16599 | Vehicle | A | 4 | **18** | 846 |
| Fried | B | 5 | 9 | 40768 | Vowel | A | 11 | 10 | 528 |
| Glass | A | 6 | 9 | 214 | Wine | A | 3 | **13** | 178 |
| Housing | B | 6 | 6 | 506 | Wisconsin | B | 16 | **16** | 194 |

**Table 2.** Experimental results of MLP-LR and their ranks

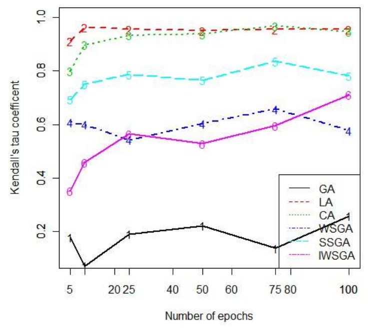
| Dataset | GA | | LA | | CA | | WSGA | | SSGA | | IWSGA | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\tau$ | $r_i^j$ | $\tau$ | $r_i^j$ | $\tau$ | $r_i^j$ | $\tau$ | $r_i^j$ | $\tau$ | $r_i^j$ | $\tau$ | $r_i^j$ |
| Authorship | 0.291 | 6 | 0.889 | 1 | 0.829 | 2 | 0.307 | 5 | 0.528 | 4 | 0.548 | 3 |
| Bodyfat | -0.004 | 5 | 0.056 | 2 | 0.075 | 1 | 0.033 | 3 | 0.022 | 4 | -0.006 | 6 |
| Calhousing | 0.054 | 6 | 0.083 | 3 | 0.106 | 2 | 0.078 | 4 | 0.130 | 1 | 0.076 | 5 |
| Cpu-small | 0.109 | 6 | 0.295 | 2 | 0.357 | 1 | 0.176 | 5 | 0.293 | 3 | 0.181 | 4 |
| Elevators | 0.110 | 6 | 0.687 | 1 | 0.684 | 2 | 0.135 | 5 | 0.419 | 3 | 0.168 | 4 |
| Fried | -0.002 | 6 | 0.532 | 2 | 0.660 | 1 | 0.157 | 4 | 0.446 | 3 | 0.133 | 5 |
| Glass | 0.317 | 5 | 0.818 | 1 | 0.757 | 2 | 0.258 | 6 | 0.475 | 4 | 0.493 | 3 |
| Housing | 0.077 | 6 | 0.531 | 2 | 0.574 | 1 | 0.290 | 3 | 0.241 | 4 | 0.094 | 5 |
| Iris | 0.178 | 6 | 0.911 | 1 | 0.800 | 2 | 0.609 | 4 | 0.693 | 3 | 0.351 | 5 |
| Pendigits | 0.161 | 5 | 0.694 | 2 | 0.752 | 1 | 0.122 | 6 | 0.314 | 3 | 0.257 | 4 |
| Segment | 0.177 | 6 | 0.799 | 2 | 0.842 | 1 | 0.341 | 4 | 0.338 | 5 | 0.346 | 3 |
| Stock | 0.032 | 6 | 0.732 | 2 | 0.745 | 1 | 0.303 | 4 | 0.403 | 3 | 0.197 | 5 |
| Vehicle | 0.106 | 6 | 0.801 | 1 | 0.800 | 2 | 0.482 | 4 | 0.504 | 3 | 0.339 | 5 |
| Vowel | 0.065 | 6 | 0.474 | 2 | 0.545 | 1 | 0.098 | 5 | 0.130 | 3 | 0.125 | 4 |
| Wine | 0.324 | 6 | 0.931 | 1 | 0.874 | 2 | 0.503 | 4 | 0.598 | 3 | 0.341 | 5 |
| Wisconsin | 0.007 | 6 | 0.221 | 2 | 0.235 | 1 | 0.066 | 3 | 0.060 | 4 | 0.028 | 5 |
| $R_j$ | 5.8125 | | 1.6875 | | 1.4375 | | 4.3125 | | 3.3125 | | 4.4375 | |

by normalizing all attributes, and separating the dataset into a training and a test set. On each dataset we tested the six approaches with $h = 3$ hidden neurons, $\eta = 0.2$, using 5 epochs with 5 random restarts. The error estimation methodology is 10-fold cross-validation. The results are presented in terms of the similarity between the rankings $\pi_i$ and $\hat{\pi}_i$ with the Kendall $\tau$ coefficient, which is equivalent to the error measure described in Section 2.

In Table 2, we show the resulting $\tau$-values for each approach, and associated rank (lower is better) per dataset. The bottom row shows the average rank for each approach, which allows us to compare the relative performance of the approaches using the Friedman test with post-hoc Nemenyi test [13]. The Friedman test proves that the average ranks are significantly unequal (with $\alpha = 1\%$). Then the Nemenyi test gives us a critical difference of $CD = 2.225$ (with $\alpha = 1\%$).



(a) Boxplot of the results according to the approaches



(b) Results per number of epochs on Iris dataset

**Fig. 1.** Results of Kendall's $\tau$ correlation coefficient

The test implies that for each pair of approaches $A_i$ and $A_j$, if $R_i < R_j - CD$, then $A_i$ is significantly better than $A_j$. Hence we can see from the table that approaches $LA$ and $CA$ significantly outperform all other approaches except for $SSGA$. However, at $\alpha = 10\%$ the critical difference becomes $CD = 1.712$, so at this significance level $CA$ significantly outperforms $SSGA$ too.

As we can see from Table 2, not all approaches have a very high $\tau$-value for all datasets. Notice, however, that these experiments are performed with a rather arbitrary set of parameters. Varying parameters such as the number of hidden neurons in the MLP, the number of epochs used when learning the neural network, and the number of random restarts, could benefit performance. To illustrate this, Figure 1b displays the variation of $\tau$-values for the different approaches on the *Iris* dataset, when varying the number of epochs. As we can see, we can subtantially improve the results when tweaking the number of epochs. For some approaches using more epochs is better, but for others this monotonicity does not hold. We see similar behavior when varying the number of stages and hidden neurons. Hence, we expect that much better results can be gained with the new approaches when the parameter space is properly explored for each dataset, but this is beyond the scope of this paper.

In Table 3, we compare the performance of approaches $LA$ and $CA$ with published results of the state-of-the-art algorithms *equal width apriori label ranking* (EW), *minimum entropy apriori label ranking* (ME) [10], *constraint classification* (CC), *instance-based label ranking* (IBLR) and *ranking trees* (LRT) [8, 10], in terms of Kendall's $\tau$ coefficient. Notice that the new methods do not generally outperform the current state-of-the-art methods, but they do achieve results that are often of the same magnitude. Since the results for the new approaches are obtained without any form of parameter optimization, we feel confident that exploration of the parameter space can yield a competitive algorithm.

**Table 3.** Comparison of MLP-LR with other methods

| Dataset | MLP-LR | | | | APRIORI-LR | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CA | $r_i^j$ | LA | $r_i^j$ | EW | $r_i^j$ | ME | $r_i^j$ | CC | $r_i^j$ | IBLR | $r_i^j$ | LRT | $r_i^j$ |
| **Authorship** | 0.829 | 5 | 0.889 | 3 | NA | NA | 0.608 | 6 | 0.920 | 2 | 0.936 | 1 | 0.882 | 4 |
| Bodyfat | 0.074 | 5 | 0.056 | 7 | 0.161 | 3 | 0.059 | 6 | 0.281 | 1 | 0.248 | 2 | 0.117 | 4 |
| Calhousing | 0.106 | 6 | 0.083 | 7 | 0.139 | 5 | 0.291 | 3 | 0.250 | 4 | 0.351 | 1 | 0.324 | 2 |
| Cpu-small | 0.357 | 5 | 0.295 | 6 | 0.279 | 7 | 0.439 | 4 | 0.475 | 2 | 0.506 | 1 | 0.447 | 3 |
| Elevators | 0.684 | 5 | 0.687 | 4 | 0.623 | 7 | 0.643 | 6 | 0.768 | 1 | 0.733 | 3 | 0.760 | 2 |
| Fried | 0.660 | 6 | 0.532 | 7 | 0.676 | 5 | 0.774 | 4 | 0.999 | 1 | 0.935 | 2 | 0.890 | 3 |
| Glass | 0.757 | 7 | 0.818 | 5 | 0.794 | 6 | 0.871 | 2 | 0.846 | 4 | 0.865 | 3 | 0.883 | 1 |
| Housing | 0.574 | 6 | 0.531 | 7 | 0.577 | 5 | 0.758 | 2 | 0.660 | 4 | 0.745 | 3 | 0.797 | 1 |
| Iris | 0.800 | 7 | 0.911 | 4 | 0.883 | 5 | 0.960 | 2 | 0.836 | 6 | 0.966 | 1 | 0.947 | 3 |
| **Pendigits** | 0.752 | 4 | 0.694 | 5 | 0.684 | 6 | NA | NA | 0.903 | 3 | 0.944 | 1 | 0.935 | 2 |
| **Segment** | 0.842 | 4 | 0.799 | 6 | 0.496 | 7 | 0.829 | 5 | 0.914 | 3 | 0.959 | 1 | 0.949 | 2 |
| Stock | 0.745 | 5 | 0.732 | 7 | 0.836 | 4 | 0.890 | 3 | 0.737 | 6 | 0.927 | 1 | 0.895 | 2 |
| **Vehicle** | 0.800 | 5 | 0.801 | 4 | 0.675 | 7 | 0.774 | 6 | 0.855 | 2 | 0.862 | 1 | 0.827 | 3 |
| Vowel | 0.545 | 6 | 0.474 | 7 | 0.709 | 3 | 0.680 | 4 | 0.623 | 5 | 0.900 | 1 | 0.794 | 2 |
| **Wine** | 0.874 | 6 | 0.931 | 3 | 0.910 | 4 | 0.844 | 7 | 0.933 | 2 | 0.949 | 1 | 0.882 | 5 |
| **Wisconsin** | 0.235 | 5 | 0.221 | 6 | 0.280 | 4 | 0.031 | 7 | 0.629 | 1 | 0.506 | 2 | 0.343 | 3 |

To learn more about our results, we crafted a metalearning dataset from Tables 1 and 3. We performed a Subgroup Discovery [14, 15] run using the dataset characteristics from Table 1 as search space, and mined for local patterns wherein the rank of LA or CA deviates from the average over all datasets. Such a run results in a set of conditions on dataset characteristics, under which our approaches perform unusually good or bad, giving pointers for further research.

The most convincing metasubgroup under which both LA and CA perform well, is defined by $m \geq 13$. Datasets belonging to this subgroup are indicated by bold blue names in Table 3. When the dataset at hand has relatively many attributes, our approaches have relatively many input signals in the MLP. Hence there are many more connections with the hidden layer, and much more interactions between the neurons in the network. Apparently, this increased complexity of the MLP adds subtlety to its predictions, which allows the MLP-LR method to induce more accurate representations of the underlying concepts.

## 5 Conclusions

Empirical results indicate that the two methods that directly incorporate the individual errors perform significantly better than the methods that focus on the LR error. However, the best results are obtained by combining both errors (CA). A comparison with results published for other methods additionally indicates that our method has the potential to compete with other methods. This holds even though no parameter tuning was carried out, which is known to be essential for learning accurate networks. Our method becomes more competitive when the data contains more attributes; this increases the amount of input neurons, and the MLP-LR predictions benefit from the more complex network. As future work, apart from parameter tuning we will investigate other ways of combining the local and global errors and we will investigate how to give more importance to higher ranks.

**Acknowledgments.** This research is financially supported by the Netherlands Organisation for Scientific Research (NWO) under project number 612.065.822 (Exceptional Model Mining) and by FCT project Rank! (PTDC/EIA/81178/2006).

## References

1. Aiguzhinov, A., Soares, C., Serra, A.P.: A Similarity-Based Adaptation of Naive Bayes for Label Ranking: Application to the Metalearning Problem of Algorithm Recommendation. In: Discovery Science (2010)
2. Vembu, S., Gärtner, T.: Label Ranking Algorithms: A Survey. In: Fürnkranz, J., Hüllermeier, E. (eds.) Preference Learning. Springer (2010)
3. Hüllermeier, E., Fürnkranz, J.: On loss functions in label ranking and risk minimization by pairwise learning. JCSS 76(1), 49–62 (2010)
4. Kanda, J., Carvalho, A.C.P.L.F., Hruschka, E.R., Soares, C.: Using Meta-learning to Classify Traveling Salesman Problems. In: SBRN (2010)
5. Brinker, K., Hüllermeier, E.: Label Ranking in Case-Based Reasoning. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 77–91. Springer, Heidelberg (2007)
6. Dekel, O., Manning, C.D., Singer, Y.: Log-linear models for label ranking. In: Advances in Neural Information Processing Systems (2003)
7. Hülermeier, E., Fürnkranz, J., Cheng, W., Brinker, K.: Label ranking by learning pairwise preferences. Artif. Intell., 1897–1916 (2008)
8. Cheng, W., Dembczynski, K., Hüllermeier, E.: Label Ranking Methods based on the Plackett-Luce Model. In: ICML (2010)
9. Cheng, W., Huhn, J.C., Hüllermeier, E.: Decision tree and instance-based learning for label ranking. In: ICML (2009)
10. de Sá, C.R., Soares, C., Jorge, A.M., Azevedo, P., Costa, J.: Mining Association Rules for Label Ranking. In: Huang, J.Z., Cao, L., Srivastava, J. (eds.) PAKDD 2011, Part II. LNCS, vol. 6635, pp. 432–443. Springer, Heidelberg (2011)
11. Haykin, S.: Neural Networks: a comprehensive foundation, 2nd edn (1998)
12. KEBI Data Repository, `http://www.uni-marburg.de/fb12/kebi/research/repository`
13. Demăr, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
14. Klösgen, W.: Subgroup Discovery. In: Handbook of Data Mining and Knowledge Discovery, ch. 16.3. Oxford University Press, New York (2002)
15. Pieters, B.F.I., Knobbe, A., Džeroski, S.: Subgroup Discovery in Ranked Data, with an Application to Gene Set Enrichment. In: Proc. Preference Learning Workshop (PL 2010) at ECML PKDD (2010)