A Probabilistic Graphical Model-Based Approach for the Label Ranking Problem

Juan Carlos Alfaro(✉), Enrique González Rodrigo, Juan Ángel Aledo, and José Antonio Gámez

School of Computer Science and Engineering, University of Castilla-la Mancha, 02071 Albacete, Spain
{JuanCarlos.Alfaro,Enrique.GRodrigo,JuanAngel.Aledo,Jose.Gamez}@uclm.es

Abstract. The goal of the *Label Ranking (LR) Problem* is to learn *preference models* that predict the preferred ranking of class labels for a given unlabelled instance. Different well-known machine learning algorithms have been adapted to deal with the LR problem. In particular, fine-tuned instance-based algorithms have exhibited a remarkable performance, specially when the model is trained with *complete* rankings, while model-based algorithms (e.g. decision trees) have been proved to be more robust when some data is missing, that is, the model is trained with *incomplete* rankings.

Probabilistic Graphical Models (*PGMs*, e.g. *Bayesian networks*) have not been considered to deal with this problem because of the difficulty to model permutations in that framework. In this paper, we propose a *Hidden Naive Bayes classifier* (*HNB*) to cope with the LR problem. By introducing the hidden variable we can design a hybrid Bayesian network in which several types of distributions can be combined, in particular, the *Mallows* distribution, which is a well-known distribution to deal with permutations. The experimental evaluation shows that the HNB classifier is competitive in accuracy when compared with *Label Ranking* (*decision*) *Trees*, being, moreover, considerably faster.

Keywords: Naive Bayes · Label Ranking · Machine learning

1 Introduction

Preferences are comparative judgments about a set of alternatives or choices. The *Label Ranking (LR) Problem* [9] is a well-known non standard supervised classification problem [7,17], whose goal is to learn *preference models* that predict the preferred ranking over a set of class labels for a given unlabelled instance.

Formally, we consider a problem domain defined over n *predictive variables* or *attributes*, $X_1, \ldots, X_n$, and a *class variable* C with k labels, $dom(C) = \{c^1, \ldots, c^k\}$. We are interested in predicting the ranking π of the labels for a given unlabelled instance $x = (x_1, \ldots, x_n) \in dom(X_1) \times \cdots \times dom(X_n)$ from a dataset $\mathbf{D} = \{(x_1^j, \ldots, x_n^j, \pi^j)\}_{j=1}^N$ with N labelled instances. Hence, the LR

G. Kern-Isberner and Z. Ognjanović (Eds.): ECSQARU 2019, LNAI 11726, pp. 351–362, 2019.
https://doi.org/10.1007/978-3-030-29765-7_29

problem consists in learning a LR-Classifier $\mathcal{C}$ from $\mathbf{D}$ which generalized well on unseen data. In other words, the goal of the LR problem is to induce a model able to predict permutations by taking advantage of all the available information in the learning process. In the literature we can find different approaches to tackle this problem:

– *Transformation methods.* They transform the whole problem into a set of single-class classifiers: *labelwise* [29] and *pairwise* approaches [15,19], *chain classifiers* [16], etc.
– *Adaptation methods.* They adapt well-known machine learning algorithms to cope with the new class structure. Cheng et al. in [9] introduce a *model-based algorithm* that induces a decision tree and a *model-free algorithm* which uses k-nearest neighbors. Other techniques, like association rules [26] or neural networks [25], have been also adapted.
– *Ensemble methods.* Recently, different tree-based aggregation approaches like *Random Forests* [5] and *Bagging predictors* [4] have been successfully applied to the LR problem [1,28,30].

In this paper we propose a new model-based LR-classifier focusing on adaptation methods. Our motivation is twofold:

– Although fine-tuned instance-based algorithms have exhibited a remarkable performance, specially when the model is trained with *complete rankings* (i.e., *permutations*), model-based algorithms have been proved to be more robust when some data is missing, that is, when the model is trained with *incomplete rankings*.
– *Probabilistic Graphical Models* (*PGMs*), e.g. *Bayesian networks* [22], have not been used in this problem because of the difficulty to model permutations in this framework [8,9].

The proposed LR-classifier is modelled by using a *hybrid Bayesian network* [12] where different probability distributions are used to conveniently model variables of distinct nature: multinomial for discrete variables, Gaussian for numerical variables and *Mallows model* for permutations [24]. The Mallows probability distribution is usually considered for permutations, and is, in fact, the core of the *decision tree algorithm* (*Label Ranking Trees*, *LRT*) proposed in [9].

To overcome the constraints regarding the topology of the network when dealing with different types of variables, we propose a *Naive Bayes* structure where the root is a *hidden discrete variable.* In that way, only univariate probability distributions have to be estimated for each state of the hidden variable. We design a learning algorithm based on the well-known *Expectation-Maximization* (EM) estimation principle and we provide several inference schemes which combine methods to tackle the *Kemeny Ranking Problem* (*KRP*) [21] with probabilistic inference.

The rest of the paper is structured as follows. In Sect. 2 we review some basic notions needed to deal with rank data. In Sect. 3 we formally describe the proposed *Hidden Naive Bayes* (*HNB*) as well as the algorithms to induce it from data and carry out inference. In Sect. 4 we set forth the empirical study carried out to evaluate the method designed in this paper. Finally, in Sect. 5, we provide the conclusions and future research lines.

2 Background

In this section, we review some notions regarding the *Kemeny Ranking Problem* [21], the *Mallows probability distribution* [24] and the *Naive Bayes model* [22].

2.1 Kemeny Ranking Problem

The *Kemeny Ranking Problem* (*KRP*) [21] consists in obtaining the *consensus permutation* (*mode*) $\pi_0 \in \mathbb{S}_k$ that best represents a sample with N permutations $\Pi = \{\pi_1, \ldots, \pi_N\}$, $\pi_i \in \mathbb{S}_k$. Here, $\mathbb{S}_k$ stands for the set of permutations of k elements. Formally, the KRP looks for the consensus permutation $\pi_0 \in \mathbb{S}_k$ that minimizes

$$\pi_0 = \underset{\pi \in \mathbb{S}_k}{argmin} \sum_{i=1}^{N} D(\pi_0, \pi_i)$$

where $D(\pi, \tau)$, $\pi, \tau \in \mathbb{S}_k$, is a distance measure between two permutations π and τ. Normally, the *Kendall distance* is taken, and the (greedy) *Borda count algorithm* [3] is employed to solve the KRP, due to its trade-off between efficiency and accuracy. Borda count algorithm basically assign n points to the item ranked first, $n-1$ to the second one and so on. Once all the input rankings have been computed, the items are sorted according to the number of accumulated points.

2.2 Mallows Probability Distribution

The *Mallows probability distribution* (*Mallows model*) [24] is an exponential probability distribution over permutations based on distances. The Mallows model is parametrized by two parameters, the *central permutation* (*mode*) $\pi_0 \in \mathbb{S}_k$ and the *spread parameter* (*dispersion*) $\theta \in [0, +\infty)$. Given a distance D in $\mathbb{S}_k$, the probability assigned to a permutation $\pi \in \mathbb{S}_k$ by a Mallows distribution with $\pi_0 \in \mathbb{S}_k$ and $\theta \in [0, +\infty)$ is

$$P(\pi; \pi_0, \theta) = \frac{e^{-\theta \cdot D(\pi, \pi_0)}}{\Psi(\theta)}$$

where $\Psi(\theta)$ is a normalization constant. The spread parameter θ quantifies the concentration of the distribution around π_0. For $\theta = 0$, a uniform distribution is obtained, while for $\theta = +\infty$, the model assigns a probability equal to 1 to π_0 and equal to 0 to the rest of the permutations. In our work, we take D as the Kendall distance.

Parameter estimation can be done by using Borda count method for π_0 and, although there is no closed form to estimate θ, numerical algorithms, e.g. Newton-Raphson, can be used to accurately estimate it. Therefore, both parameters can be efficiently estimated (polinomial time) [20].

2.3 Naïve Bayes

Naive Bayes models [22] are well-known *probabilistic classifiers* based on the conditional independence hypothesis, that is, every pair of features are considered conditionally independent given the class variable. As most of the probabilistic classifiers, Naive Bayes models follow the *maximum a posteriori* (*MAP*) principle, that is, they return the most probable class for any input instance. Formally, given an input instance $x = (x_1, \dots x_n) \in dom(X_1) \times \cdots \times dom(X_n)$ and being C the class variable with $dom(C) = \{c^1, \dots, c^n\}$, a *Naive Bayes Classifier* $\mathcal{C}$ returns

$$\mathcal{C}(x) = \underset{c \in dom(C)}{argmax} P(c\,|x) = \underset{c \in dom(C)}{argmax} P(x, c) = \underset{c \in dom(C)}{argmax} \prod_{i=1}^{n} P(x_i|c) \cdot P(c)$$

according to the *Bayes' theorem* and the conditional independence hypothesis, respectively. The conditional distributions above may be multinomial for discrete attributes and Gaussian for continuous attributes.

3 Hidden Naïve Bayes LR-Classifier

This section presents the proposed model, defining the structure as well as the parameter estimation process.

3.1 Model Definition

To overcome the constraints regarding the topology of the network when dealing with different types of variables, the model proposed here is a mixture model with the Naive Bayes assumption. The root element of the model is the discrete hidden variable, which we will denote as $z \in 1...Z$, where Z is the total number of mixture models. The rest of the variables are observed variables. We consider two types of observed variables:

– The *feature variables*, observed both in the training and in the test phase. We consider two kinds: discrete variables, denoted as d_j, and continuous variables, denoted as x_k.
– The *ranking variable*, denoted as π, which is only present at training time. This is the one to infer.

Figure 1 offers a representation of the model with the different types of variables described above. The model assumes that each of these variable types follow a different conditional distribution given the root variable:

– Continuous variables follow a Gaussian distribution, $P(x_k|z) = \mathcal{N}(x_k; \mu_k^z, \sigma_k^z)$
– Discrete variables follow a Multinomial distribution, $P(d_j|z) = Mult(d_j; \mathbf{a_j^z})$
– The ranking variable follows a Mallows distribution, $P(x_k|z) = \mathscr{M}(\pi; \pi_0^z, \theta^z)$
– The hidden variable follows a Multinomial distribution, $P(z) = Mult(z; \mathbf{w})$

The parameters for each of the conditional distributions need to be estimated to perform inference using the model.

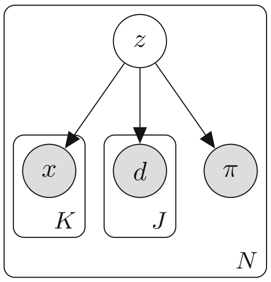

Fig. 1. The proposed Hidden Naïve Bayes model

3.2 Parameter Estimation

Due to the fact that the model has one hidden variable, we use the EM algorithm to estimate jointly the parameters of both the observed and the hidden variables. The EM algorithm consists of two steps: the Expectation step (E step), where the value for the hidden variable is estimated; and the Maximization step (M step), where the parameters for the conditional distributions are obtained.

E step. Under the assumption that the parameters of the model (μ_k^z, σ_k^z, $\mathbf{a_j^z}$, π_0^z, θ^z, $\mathbf{w}$) are known, the probability of an example to be in a mixture is

$$P(z_i|\mathbf{d_i}, \mathbf{x_i}, \pi_i) \propto P(z_i)P(\mathbf{d_i}, \mathbf{x_i}, \pi_i|z_i) = P(z)P(\pi_i|z) \prod_j P(d_{ij}|z_i) \prod_k P(x_{ik}|z_i)$$

Normalizing the above expression for all values of the hidden variable we obtain the probability of an example to be in the mixture.

M step. Under the assumption that the probabilities of belonging to each mixture for all examples are known, the parameters of the model can be estimated as follows:

– Multinomial parameters for the discrete variables, $P(d_j|z)$. MLE estimation is done, where the count for each instance is weighted by the probability of that instance given a mixture $z = z_i$.
– Gaussian distribution parameters for the continuous variables, $P(x_k|z)$. Analogous to the previous case but using a Gaussian distribution $\mathcal{N}(x_k; \mu_k^z, \sigma_k^z)$ for each $z = z_i$.
– Mallows distribution parameters for the ranking variable. For each $z = z_i$ a Mallows distribution $\mathscr{M}(\pi; \pi_0^z, \theta^z)$ must be estimated. In particular $\pi_0^{z_i}$ is computed by applying a weighted version of Borda count algorithm (points assigned to items are weighted by the probability of that instance given the mixture z_i), and θ^{z_i} is calculated by using a numerical optimization process (e.g. Newton-Raphson).
– The mixture model probabilities $P(z)$ are computed according to the weights $P(z|\mathbf{d_i}, \mathbf{x_i}, \pi_i)$ for each mixture $z = z_i$.

Stop Condition. Although the model can easily be extended to use several types of stop conditions, we propose to check the convergence on the probabilities with which the samples belong to each mixture.

3.3 Learning Process

The learning procedure includes several executions of the EM algorithm with an increasing number of mixtures. This process is based on the one proposed in [23]. The algorithm starts with a predefined number of mixtures (a hyperparameter), and at each iteration the number of mixtures is increased according to a given parameter (in our case, one by one).

For each of these iterations, the mixtures must be initialized as a previous step to the EM algorithm. For each new mixture added, a sample with replacement of the dataset is used for parameter estimation. The parameters for the conditional probabilities given this mixture are calculated as if all the data points had a probability of 1 to belong to the sample. After that, the data points which were not used for the initialization of the new mixtures are used for the parameter optimization procedure. If the solution obtained does not improve upon the previous solution (using the *Kendall coefficient* τ_K as score over a validation set), the algorithm returns the best solution. If the solution improves, the algorithm continues adding new mixtures.

3.4 Inference

In the inference process, the method needs to predict the best consensus ranking for a new data point. In our proposal we do that by marginalizing variables until obtaining an expression for the probability of a ranking

$$P(\pi_s|\mathbf{d_r}, \mathbf{x_r}) \propto \sum_{z_i} P(z_i) P(\pi_s|z_i) \prod_{j}^{J} P(d_{rj}|z_i) \prod_{k}^{K} P(x_{rk}|z_i)$$

To estimate the best permutation $\widetilde{\pi}$, we take the one that maximizes the score

$$\pi^* = \underset{\pi_s \in \mathbb{S}_k}{argmax} \; P(\pi_s|\mathbf{d_r}, \mathbf{x_r})$$

However, due to the number of values of π, an approximation may be obtained by aggregating the rankings weighted by the factor given by the marginalization

$$P(z_i|\mathbf{d_r}, \mathbf{x_r}) \propto P(z_i) \prod_{j}^{J} P(d_{rj}|z_i) \prod_{k}^{K} P(x_{rk}|z_i)$$

Then, we apply weighted Borda count by using the consensus permutation identified for each component z_i of the mixture and using probabilities $P(z_i|\mathbf{d_r}, \mathbf{x_r})$ as weights.

4 Experimental Evaluation

In this section we carry out an experimental evaluation to assess the performance of the proposed algorithm. Next, we describe the employed datasets, the tested algorithms, the methodology and the results.

4.1 Datasets

We used the 21 datasets proposed in [9,19]. The first 16 may be considered semi-synthetic since they were obtained by transforming 8 multi-class (type A) and 8 regression datasets (type B) to the LR problem, while the last 5 correspond to real-world biological problems. Table 1 provides the main characteristics of each dataset. The columns #rankings and max #rankings correspond to the number of different rankings in the dataset and the maximum number of rankings that can be generated for such dataset, respectively.

Table 1. Datasets description.

Dataset	type	#instances	#features	#labels	#rankings	max #rankings
authorship	A	841	70	4	17	4!
bodyfat	B	252	7	7	236	7!
calhousing	B	20640	4	4	24	4!
cpu-small	B	8192	6	5	119	5!
elevators	B	16599	9	9	131	9!
fried	B	40769	9	5	120	5!
glass	A	214	9	6	30	6!
housing	B	506	6	6	112	6!
iris	A	150	4	3	5	3!
pendigits	A	10992	16	10	2081	10!
segment	A	2310	18	7	135	7!
stock	B	950	5	5	51	5!
vehicle	A	846	18	4	18	4!
vowel	A	528	10	11	294	11!
wine	A	178	13	3	5	3!
wisconsin	B	194	16	16	194	16!
spo	-	2465	24	11	2361	11!
heat	-	2465	24	6	622	6!
dtt	-	2465	24	4	24	4!
cold	-	2465	24	4	24	4!
diau	-	2465	24	7	967	7!

4.2 Algorithms

The algorithms involved in the experimental evaluation were the following ones:

- The model-based *Label Ranking Trees* (*LRT*) algorithm [9], based on decision tree induction [6]. To avoid overfitting, we fixed the minimum number of instances for splitting an inner node to twice the number of class labels.
- The model-free *Instance-Based Label Ranking* (*IBLR*) algorithm [9], which follows the nearest neighbors paradigm [10]. The nearest neighbors were identified by using the Euclidean distance. The number of nearest neighbors were adjusted by applying an inner five-fold cross validation method (5-cv) over the training fold.
- Our model-based proposal of *Hidden Naive Bayes LR-classifier* (*HNB*).

4.3 Methodology

We adopted the following design decisions:

- The algorithms were assessed by using a five repetitions of a ten-fold cross validation method (5×10-cv).
- The *Kendall coefficient* τ_K was used as goodness score (see [1] for details).
- The algorithms were implemented in Python 3.6.5 and the experiments executed in computers running CentOS Linux 7 with CPU Intel(R) Xeon(R) E5-2630 running at 2.40 GHz and 16 GB of RAM memory.

4.4 Results

Next, we provide the accuracy and time results, as well as their corresponding statistical analysis.

The accuracy results are shown in Table 2. Each cell contains the mean and the standard deviation of the Kendall coefficient τ_K for the test folds over the 5×10-cv. The boldfaced cells correspond to the algorithm(s) that obtain(s) the best result for each dataset.

To properly analyze the results, we applied the standard statistical analysis procedure for machine learning algorithms described in [11,14] by using the `exreport` package [2]. This procedure can be divided in two steps:

- First, a *Friedman test* [13] was applied using a significance level of $\alpha = 0.05$. The obtained $p-value$ was $3.253\mathrm{e}^{-5}$, and so we rejected the null hypothesis (H_0) that all the algorithms were equivalent in terms of accuracy in favour of the alternative (H_1), that is, at least one of them was different.
- Second, taking as control the algorithm ranked first by the Friedman test (IBLR), we performed a post-hoc test with the *Holm's procedure* [18], also using a significance level of $\alpha = 0.05$. This test compares all the algorithms with the one taking as control to discover the outstanding methods. The results for the post-hoc test are shown in Table 3. The *win*, *tie* and *loss* columns stand for the number of datasets in which the control algorithm wins, ties and losses with respect to the one on the column *Method*.

According to these results, we can conclude that:

- The Friedman test ranked first the IBLR algorithm, which was taken as control for the post-hoc test. LRT was ranked second, and HNB third.
- The post-hoc test revealed that the IBLR algorithm was statistically different in terms of accuracy with respect to HNB and LRT.
- Regarding LRT and HNB, the pairwise Shaffer's post-hoc test [27] ($\alpha = 0.05$) obtained a $p\!-\!value$ of $8.774e^{-1}$. Therefore, we can not reject the null hypothesis (H_0) that these algorithms were equivalent in terms of accuracy.

Table 2. Accuracy results for each algorithm.

Dataset	IBLR	LRT	HNB
aut	**0.932 (± 0.013)**	0.862 (± 0.033)	0.918 (± 0.018)
bod	**0.224 (± 0.067)**	0.159 (± 0.070)	0.116 (± 0.073)
cal	0.337 (± 0.010)	**0.340 (± 0.011)**	0.183 (± 0.026)
cpu	**0.501 (± 0.013)**	0.445 (± 0.015)	0.429 (± 0.014)
ele	0.728 (± 0.007)	**0.753 (± 0.008)**	0.683 (± 0.021)
fri	0.975 (± 0.001)	0.893 (± 0.003)	0.747 (± 0.120)
gla	**0.838 (± 0.072)**	0.829 (± 0.064)	0.837 (± 0.077)
hou	0.721 (± 0.0339)	**0.757 (± 0.033)**	0.418 (± 0.255)
iri	0.955 (± 0.042)	0.924 (± 0.056)	**0.958 (± 0.044)**
pen	**0.941 (± 0.002)**	0.924 (± 0.003)	0.863 (± 0.006)
seg	**0.951 (± 0.006)**	0.943 (± 0.007)	0.773 (± 0.055)
sto	**0.921 (± 0.011)**	0.894 (± 0.018)	0.888 (± 0.018)
veh	**0.854 (± 0.027)**	0.811 (± 0.044)	0.805 (± 0.039)
vow	**0.870 (± 0.016)**	0.718 (± 0.037)	0.748 (± 0.039)
win	**0.945 (± 0.039**)	0.885 (± 0.071)	0.934 (± 0.049)
wis	**0.491 (± 0.047)**	0.373 (± 0.046)	0.386 (± 0.051)
spo	**0.148 (± 0.017)**	0.105 (± 0.016)	0.144 (± 0.018)
hea	**0.061 (± 0.024)**	0.035 (± 0.020)	0.052 (± 0.022)
dtt	**0.127 (± 0.032)**	0.075 (± 0.038)	0.117 (± 0.034)
col	**0.076 (± 0.028)**	0.051 (± 0.027)	0.065 (± 0.034)
dia	**0.225 (± 0.027)**	0.151 (± 0.025)	0.217 (± 0.028)

Table 3. Post-hoc test for the accuracy results.

Method	Rank	$p\!-\!value$	Win	Tie	Loss
IBLR	1.19	-	-	-	-
LRT	2.38	$1.205e^{-4}$	18	0	3
HNB	2.43	$1.205e^{-4}$	20	0	1

In this work, we deal with model-free and model-based methods for the LR problem, whose CPU requirements are clearly different. Therefore, to make a fair comparison, the time for the whole process (learning with the training dataset and validating with the test one) was gathered. The improvement ratios (time) of HNB with respect to IBLR and LRT are shown in Table 4.

Table 4. Time results for each algorithm.

Method	aut	bod	cal	cpu	ele	fri	gla	hou	iri	pen
IBLR	2.007	1.664	74.086	35.242	60.933	444.556	0.535	2.485	0.428	24.819
LRT	11.477	0.868	1.222	1.213	2.237	4.169	2.210	1.042	0.086	1.729

Method	seg	sto	veh	vow	win	wis	spo	hea	dtt	col	dia
IBLR	8.704	1.834	1.820	1.216	0.864	0.874	23.612	11.223	17.486	10.410	24.706
LRT	15.667	1.024	1.024	2.061	6.081	2.515	14.479	8.724	9.012	8.334	10.997

In light of these results, we can highlight that the HNB classifier is two times faster than the IBLR algorithm for smaller datasets, while for larger ones this value is multiplied by a factor of ten. Regarding the LRT method, we observe that the HNB classifier is two and ten times faster for smaller and larger datasets, respectively. Therefore, we may sacrifice a bit of time to improve the parameters of the HNB model.

Finally, it should be remarked that there are some datasets (e.g., segment or pendigits) where the HNB model fails in the prediction task when compared with the IBLR and LRT algorithms. To find an explanation, we decided to apply the corresponding algorithms for some of these datasets but in the classification setup. When examining these results, we realized that the Naive Bayes algorithm also failed, while decision trees and instance-based methods succeeded. Thus, we think that the assumption that all the features follow a Gaussian distribution restrict the predictive power of PGMs. Our suspicions were confirmed when we observed that the results of the Naive Bayes model strongly improved when the features were discretized. Therefore, we expect that the HNB model also improves when we properly apply multinomial probability distributions instead of Gaussian ones.

5 Conclusions

In this paper, we cope with the LR problem. Based on the EM estimation principle, we have defined a Naive Bayes structure where the root is a hidden discrete variable, used to model the different probability distributions that must be managed in such problem (multinomial and Gaussian for the features and Mallows for the permutations).

From the experimental evaluation, we can conclude that our proposal of Hidden Naive Bayes is clearly faster than the LRT and IBLR methods while being competitive in accuracy with the first one.

As future research we plan to introduce a discretization method to treat as discrete variables those features that does not follow a Gaussian probability distribution. Also, we will deal with a more general approach where incomplete rankings are allowed on the training dataset.

Acknowledgements. This work has been partially funded by FEDER funds, the Spanish Government (AEI/MINECO) through the project TIN2016-77902-C3-1-P and the Regional Government (JCCM) by SBPLY/17/180501/000493.

References

1. Aledo, J., Gámez, J., Molina, D.: Tackling the supervised label ranking problem by bagging weak learners. Inf. Fusion **35**, 38–50 (2017)
2. Arias, J., Cózar, J.: ExReport: fast, reliable and elegant reproducible research (2015). http://exreport.jarias.es/
3. Borda, J.: Memoire sur les elections au scrutin. Histoire de l'Academie Royal des Sciences (1770)
4. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)
5. Breiman, L.: Random forests. Mach. Learn. **45**, 5–32 (2001)
6. Breiman, L., Friedman, J., Stone, C., Olshen, R.: Classification and Regression Trees. Wadsworth Inc., Wadsworth (1984)
7. Charte, D., Charte, F., García, S., Herrera, F.: A snapshot on nonstandard supervised learning problems: taxonomy, relationships, problem transformations and algorithm adaptations. Prog. Artif. Intell. **8**(1), 1–14 (2019)
8. 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, pp. 215–222 (2010)
9. Cheng, W., Hühn, J., Hüllermeier, E.: Decision tree and instance-based learning for label ranking. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 161–168. ACM (2009)
10. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13**, 21–27 (1967)
11. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
12. Fernández, A., Gámez, J.A., Rumí, R., Salmerón, A.: Data clustering using hidden variables in hybrid Bayesian networks. Prog. Artif. Intell. **2**, 141–152 (2014)
13. Friedman, M.: A comparison of alternative tests of significance for the problem of m rankings. Ann. Math. Stat. **11**, 86–92 (1940)
14. Garcša, S., Herrera, F.: An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. J. Mach. Learn. Res. **9**, 2677–2694 (2008)
15. Gurrieri, M., Fortemps, P., Siebert, X.: Alternative decomposition techniques for label ranking. In: Laurent, A., Strauss, O., Bouchon-Meunier, B., Yager, R.R. (eds.) IPMU 2014. CCIS, vol. 443, pp. 464–474. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08855-6_47

16. Har-Peled, S., Roth, D., Zimak, D.: Constraint classification for multiclass classification and ranking. In: Proceedings of the 15th International Conference on Neural Information Processing Systems, pp. 785–792 (2002)
17. Hernández, J., Inza, I., Lozano, J.A.: Weak supervision and other non-standard classification problems: a taxonomy. Pattern Recogn. Lett. **69**, 49–55 (2016)
18. Holm, S.: A simple sequentially rejective multiple test procedure. Scand. J. Stat. **6**, 65–70 (1979)
19. Hüllermeier, E., Fürnkranz, J., Cheng, W., Brinker, K.: Label ranking by learning pairwise preferences. Artif. Intell. **172**, 1897–1916 (2008)
20. Irurozk, E., Calvo, B., Lozano, J.A.: PerMallows: an R package for mallows and generalized mallows models. J. Stat. Softw. **71**(12), 1–30 (2016)
21. Kemeny, J., Snell, J.: Mathematical Models in the Social Sciences. MIT Press, Cambridge (1972)
22. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning. MIT Press, Cambridge (2009)
23. Lowd, D., Domingos, P.: Naive Bayes models for probability estimation. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 529–536. ACM (2005)
24. Mallows, C.L.: Non-null ranking models. Biometrika **44**, 114–130 (1957)
25. Ribeiro, G., Duivesteijn, W., Soares, C., Knobbe, A.: Multilayer perceptron for label ranking. In: Villa, A.E.P., Duch, W., Érdi, P., Masulli, F., Palm, G. (eds.) ICANN 2012. LNCS, vol. 7553, pp. 25–32. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33266-1_4
26. 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. LNCS (LNAI), vol. 6635, pp. 432–443. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20847-8_36
27. Shaffer, J.P.: Multiple hypothesis testing. Annu. Rev. Psychol. **46**(1), 561–584 (1995)
28. de Sá, C.R., Soares, C., Knobbe, A.J., Cortez, P.: Label ranking forests. Expert Systems **34**(1), e12166 (2017)
29. Cheng, W., Henzgen, S., Hüllermeier, E.: Labelwise versus pairwise decomposition in label ranking. In: Proceedings of the Workshop on Lernen, Wissen & Adaptivität, pp. 129–136 (2013)
30. Zhou, Y., Qiu, G.: Random forest for label ranking. Expert Syst. Appl. **112**, 99–109 (2018)