Mach Learn (2013) 93:191–225
DOI 10.1007/s10994-013-5374-3

Supervised clustering of label ranking data using label preference information

Mihajlo Grbovic · Nemanja Djuric · Shengbo Guo · Slobodan Vucetic

Received: 10 January 2012 / Accepted: 29 April 2013 / Published online: 7 June 2013

Abstract This paper studies supervised clustering in the context of label ranking data. The goal is to partition the feature space into K clusters, such that they are compact in both the feature and label ranking space. This type of clustering has many potential applications. For example, in target marketing we might want to come up with K different offers or marketing strategies for our target audience. Thus, we aim at clustering the customers' feature space into K clusters by leveraging the revealed or stated, potentially incomplete customer preferences over products, such that the preferences of customers within one cluster are more similar to each other than to those of customers in other clusters. We establish several baseline algorithms and propose two principled algorithms for supervised clustering. In the first baseline, the clusters are created in an unsupervised manner, followed by assigning a representative label ranking to each cluster. In the second baseline, the label ranking space is clustered first, followed by partitioning the feature space based on the central rankings. In the third baseline, clustering is applied on a new feature space consisting of both features and label rankings, followed by mapping back to the original feature and ranking space. The RankTree principled approach is based on a Ranking Tree algorithm previously proposed for label ranking prediction. Our modification starts with K random label rankings and iteratively splits the feature space to minimize the ranking loss, followed by re-calculation of the K rankings based on cluster assignments. The MM-PL approach is a

Editors: Eyke Hüllermeier and Johannes Fürnkranz.

M. Grbovic (✉) · N. Djuric · S. Vucetic
Department of Computer and Information Sciences, Center for Data Analytics and Biomedical Informatics, Temple University, Philadelphia, PA 19122, USA
e-mail: mihajlo.grbovic@temple.edu

N. Djuric
e-mail: nemanja.djuric@temple.edu

S. Vucetic
e-mail: slobodan.vucetic@temple.edu

S. Guo
Xerox Research Centre Europe, 6 chemin de Maupertuis, 38240 Meylan, France
e-mail: shengbo.guo@xrce.xerox.com

multi-prototype supervised clustering algorithm based on the Plackett-Luce (PL) probabilistic ranking model. It represents each cluster with a union of Voronoi cells that are defined by a set of prototypes, and assign each cluster with a set of PL label scores that determine the cluster central ranking. Cluster membership and ranking prediction for a new instance are determined by cluster membership of its nearest prototype. The unknown cluster PL parameters and prototype positions are learned by minimizing the ranking loss, based on two variants of the expectation-maximization algorithm. Evaluation of the proposed algorithms was conducted on synthetic and real-life label ranking data by considering several measures of cluster goodness: (1) cluster compactness in feature space, (2) cluster compactness in label ranking space and (3) label ranking prediction loss. Experimental results demonstrate that the proposed MM-PL and RankTree models are superior to the baseline models. Further, MM-PL is has shown to be much better than other algorithms at handling situations with significant fraction of missing label preferences.

Keywords Label ranking · Supervised clustering · Preference learning

1 Introduction

Label Ranking is emerging as an important and practically relevant field. Unlike the standard problems of classification and regression, learning of label ranking is a complex learning task, which involves prediction of strict label order relations, rather than single values. Specifically, in the label ranking scenario, each instance, which is described by a set of features $\mathbf{x}$, is assigned a ranking of labels π, that is a total (e.g., $\pi = (5, 3, 1, 4, 2)$) or partial (e.g., $\pi = (5, 3, 2)$) order over a finite set of class labels $\mathcal{Y}$ (e.g., $\mathcal{Y} = \{1, 2, 3, 4, 5\}$). The label ranking problem consists of learning a model that maps instances $\mathbf{x}$ to a total label order $f : \mathbf{x} \rightarrow \pi$. It is assumed that a sample from the underlying distribution $D = \{(\mathbf{x}_n, \pi_n), n = 1, \ldots, N\}$, where $\mathbf{x}_n$ is a d-dimensional feature vector and π_n is a vector containing a total or partial order of a finite set $\mathcal{Y}$ of L class labels, is available for training. This problem has recently received a lot of attention in the machine learning community and has been extensively studied (Dekel et al. 2003; Cheng et al. 2009, 2010; Har-Peled et al. 2003; Qin et al. 2010). A survey of recent label ranking algorithms can be found in Gärtner and Vembu (2010).

There are many practical applications in which the objective is to learn a label preference of an instance. For example, in the case of document categorization, where it is very likely that a document belongs to multiple topics (e.g., sports, entertainment, baseball, etc.), multi-label classification has traditionally been used. However, one might not be interested only in distinguishing between relevant and irrelevant topics for a specific document, but also in learning how to find a total order of the topics by relevance. Label ranking models can be trained to perform such task, given document training data that contains topic orderings or topic scores. Note that the existence of such training data is what distinguishes a label ranking problem from a multi-label classification one, as in both cases the output may be in a form of a total order of labels. Other examples of label ranking applications include training models to predict food preferences (Kamishima and Akaho 2009). Given historical data about customers' food rankings, customer demographics and other characteristics, a label ranking model that takes customer features (e.g., age, gender, etc.) as an input and predicts food rankings as an output can be trained. Similarly, label ranking can be used for ranking of tags in images by relevance (Zhuang and Hoi 2011), given training data in form of image features and top-k tag ranks for each image. Additional applications include: meta-learning

(Vilalta and Drissi 2002), where, given a new data set, the task is to induce a total rank of available algorithms according to their suitability based on the data set properties; ranking of movie suggestions for new members of a movie website based on user features; determining an order of questions in a survey for a specific user based on respondent's attributes. See Domshlak et al. (2011) for an overview of label ranking applications in economics, operations research, and databases.

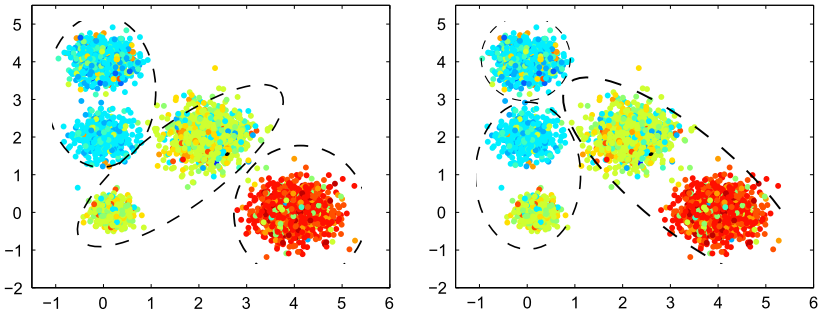
In this paper we investigate supervised clustering of label ranking data, which is an open and non-trivial problem that has not been addressed in the machine learning literature. The goal is to cluster the data based on both instance features $\mathbf{x}$ and the assigned, potentially incomplete, label rankings π, such that the label ranks of instances within a cluster are compact, meaning that they are more similar to each other than they are to the label ranks of instances in other clusters. Simultaneously, we aim at producing clusters that are compact in the feature space, such that simple rules can be used to assign new examples to clusters based on feature information only.

The problem can be formally stated as follows. Using data $D = (\mathbf{x}_i, \pi_i)$, $i = 1, \ldots, N$ the goal is to find K clusters, by learning cluster representation in the feature space $\boldsymbol{\mu}_k$ and cluster central rankings ρ_k. The k-th cluster is defined as $S_k = (\boldsymbol{\mu}_k, \rho_k)$, where depending on the actual clustering algorithm used, $\boldsymbol{\mu}_k$ represents a cluster centroid, a set of Voronoi cells, or any other feature abstraction, e.g., Decision Tree. Previously unobserved instance can be assigned to a cluster based on its feature vector x_u, e.g., by calculating the Euclidean distance $d_e(\mathbf{x}_u, \boldsymbol{\mu}_k)$. The output π_u can therefore take only one of K possible rankings, i.e., cluster central rankings. This clearly differentiates supervised clustering of label ranking from supervised label ranking, where the output is not limited to a set of K rankings. Instead, the cardinality of the output space is $L!$.

To evaluate the clustering solution, we use both internal and external clustering measures. The external measures employ ground truth about rankings of cluster members, while the internal measures just consider the cluster features. To measure the within-cluster compactness of label rankings we propose to use a measure based on the average Kendall tau distance. To measure the compactness in the feature space we use several standard internal cluster goodness measures.

One can envision many potential applications of supervised clustering for label ranking data. For example, in customer segmentation and targeted marketing applications, a company with several products would like to cluster its customers based on their product preferences for purposes of designing cluster-specific promotional material. It is assumed that the company has access to customer product preferences, which are either stated in a survey or extracted from purchase history, based on which it can cluster data in feature space. For each cluster, the company can make a different catalog, by promoting a selected subset of products and designing a catalog in a way that best reflects the taste of its target customers. Another application would be how to order the questions in K versions of the same survey to be mailed to specific target groups, given historical data about people that filled out the survey on-line, in such manner that they were allowed to choose the order of questions they answer, i.e., which questions to answer first and which to leave for last or not to answer.

Traditionally, clustering techniques are unsupervised. The term supervised clustering relates to using additional information other than instance features when performing clustering, such as class memberships (classification), target values (regression), or, in our case, label rankings. It can be found in some related literature (Eick et al. 2006, 2011; Finley and Joachims 2005), where instance class labels where used to ensure maximum cluster class purity. However, it would be very hard to apply these algorithms to the problem of label ranking directly, since treating label ranking as classification could result in having $L!$ different classes.



(a) Clustering configuration 1 (b) Clustering configuration 2

Fig. 1 Clustering of label ranking data toy example

It should be noted that unsupervised clustering of label rankings themselves has been studied before in slightly different setting (Kamishima and Akaho 2009; Meilă and Bao 2010). There it was assumed that each instance is described with a ranking π_n, and that features do not exist. As opposed to that work, we study the problem of clustering of instances in the feature space that takes into account information about label rankings. For example, if we wished to produce K marketing catalogs based on customer preferences, we could cluster the label rankings into K cluster central rankings. However, we would not have a way of assigning catalogs to new customers for whom we only have feature information, nor would we be able to examine the feature representation of clusters.

To illustrate the problem of supervised clustering of label ranking data, let us consider a toy example in Fig. 1. We show a synthetic label ranking data set with 2 features (x and y axis) and 5 labels that are ranked in a specific order for each instance. To visualize the assigned label rankings we use the following color scheme. We sort all the rankings based on the Kendall distance from reference rank $(1, 2, 3, 4, 5)$. Then colors of ranks $(1, 2, 3, 4, 5)$ and $(5, 4, 3, 2, 1)$ will be in two different ends of the color spectrum. More specifically, we set the color of the closest rank to white, color of the furthest to black, and the remaining colors are obtained by linearly interpolating between these two extremes. If the goal was to make $K = 3$ clusters, Fig. 1a and Fig. 1b show examples of potential partitions that unsupervised clustering methods would produce. Even though both configurations are equally good from the unsupervised clustering perspective, we would prefer the one in Fig. 1a, as its clusters are more compact in the label ranking space. For example, if instances correspond to customers and clusters correspond to catalogs, the partitions in Fig. 1a would make customers more inclined towards buying the products when the catalogs arrive at their doorstep. Therefore, the supervised clustering methods should aim at utilizing instance label rankings to produce partitions such as the ones in Fig. 1a.

Since supervised clustering of label ranking data, to the best of our knowledge, has not been studied before, we propose several heuristic baseline algorithms and principled models specifically tailored for this type of clustering.

The first proposed baseline approach uses a simple assumption that the clusters in feature space should correspond to clusters in the label space. For example, it assumes that all women ages 20 to 25 have similar product preferences. Therefore, in this approach, we first find K clusters in the feature space, e.g., by using the well-known K-means algorithm, followed by assigning label rankings to the obtained clusters using a generalized Mallows model (Mallows 1957) or Placket-Luce model (Plackett 1975).

In the second baseline approach we first find clusters in the label ranking space using algorithms such as those from Kamishima and Akaho (2009) and Meilă and Bao (2010), without taking the features into account. Once the cluster central rankings have been found, we establish cluster representation in the feature space. This step is needed so we could assign new examples to clusters based on their features. For example, we could keep all cluster instances in memory and use 1-nearest neighbor rule, or we could train a decision tree that assigns instances to one of the K clusters.

In the third baseline approach, we create a new feature space by adding label rankings to the original features and use standard unsupervised clustering. This is followed by processing of the obtained clusters to map clusters back to the original feature and label ranking space.

The first principled model, the Plackett-Luce Mixture Model (MM-PL) is based on a multi-prototype cluster representation, where the underlying cluster preferences are modeled using cluster-specific Plackett-Luce score parameters. In this model, the k-th cluster, $S_k = (\boldsymbol{\mu}_k, \rho_k)$, is represented in the feature space as a union of Voronoi cells and in the label ranking space using a Plackett-Luce score vector. The model is fast, with linear training and prediction time, constant memory scaling with number of prototypes and is capable of efficiently working with incomplete rankings.

The second principled model, the RankTree model (RT) for clustering is an extension of previously proposed Decision Tree algorithm for label ranking (Cheng et al. 2009). It performs supervised clustering by iteratively growing a tree to assign instances to clusters based on the current cluster central rankings, and recalculating cluster central rankings based on new assignments.

The paper is organized as follows. Section 2 describes the problem setup and the metrics against which different label ranking clustering methods can be compared. Section 3 presents heuristic baselines for this problem. Section 4 describes RankTree algorithm for supervised clustering of label ranking data. The proposed Mixture Model label ranking approach based on the PL model is covered in Sect. 5. Finally, in Sect. 7, we report results of the comprehensive experiments on both synthetic and real-world data sets, including success rates of uncovering simulated cluster central rankings in synthetic data, influence of number of clusters on accuracy and cluster compactness and influence of partial rankings on different algorithms.

2 Preliminaries

2.1 Problem setup

In the label ranking scenario, a single instance, described by a d-dimensional vector $\mathbf{x}$, is associated with a total order of assigned class labels. Specifically, we define a total order by using a transitive and asymmetric relation $\succ_{\mathbf{x}}$ on a finite set of labels $\mathcal{Y}$, where $y_i \succ_{\mathbf{x}} y_j$ indicates that label y_i precedes label y_j given $\mathbf{x}$. The total order can be represented with a permutation π of the set $\{1, \ldots, L\}$, where L is the number of available class labels. We define π such that $\pi(i)$ is the index of the class label at i-th position in the order and $\pi^{-1}(j)$ is the position of y_j in the order. The permutation can also describe incomplete rankings in form $y_{\pi(1)} \succ_{\mathbf{x}} y_{\pi(2)} \succ_{\mathbf{x}}, \ldots, \succ_{\mathbf{x}} y_{\pi(l)}$, where $l < L$ and $\{\pi(1), \ldots, \pi(l)\} \subset \{1, \ldots, L\}$.

Let us assume that N independent historical observations are collected in form of a data set $D = \{(\mathbf{x}_n, \pi_n), n = 1, \ldots, N\}$, that we wish to cluster in the feature space using assigned label ranks. The objective is to segment D into K clusters S_k, $k = 1, \ldots, K$, such that each

cluster spans a certain portion of feature space determined by some abstraction $\boldsymbol{\mu}_k$ and is associated with a certain central ranking $\rho_k, k = 1, \ldots, K$, both to be determined from D. Preferably, we would like all label rankings π_n of $\mathbf{x}_n \in S_k$ to be similar and all at a short distance from ρ_k. A new unlabeled instance $\mathbf{x}_u$ is assigned to m-th cluster by utilizing cluster feature space abstractions $\boldsymbol{\mu}_k, k = 1, \ldots, K$ and as such is assigned ranking $\hat{\pi}_k = \rho_m$.

To measure the degree of correspondence between two label ranks π_1 and π_2 it is common to use the Kendall tau distance that counts the number of discordant label pairs

$$d_\tau = \left|\left\{(y_i, y_j) : \pi_1^{-1}(y_i) > \pi_1^{-1}(y_j) \wedge \pi_2^{-1}(y_j) > \pi_2^{-1}(y_i)\right\}\right|. \tag{1}$$

Kendall tau distance is often normalized such that it lies in the interval [0,1], where 1 indicates maximum disagreement. This is done by dividing (1) by $L \cdot (L-1)/2$.

2.2 Evaluation metrics

The validation of clustering results has always been the most difficult and frustrating part of any cluster analysis (Jain and Dubes 1988). This holds for evaluation no matter whether or not a ground truth (e.g., class labels) is available against which to compare the clustering result. The evaluation measures can strictly be categorized into external and internal measures (Brun et al. 2007; Kremer et al. 2011), depending on whether or not they use ground truth in evaluation.

Evaluation of potential label ranking data clustering solutions as such is not a trivial task as well. Firstly, we would prefer the clusters to be pure in the label ranking space, thus ensuring that the central cluster rankings are good representatives of cluster members. Secondly, we would prefer the clusters to be compact in the feature space so that we can use simple and intuitive rules to divide our population. Finally, and in some applications most importantly, we would like to measure the quality of potential clustering solutions on previously unseen examples, i.e., how close is the true label rank to the one predicted on basis of cluster membership. Intuitively, this would be analogous to measuring customer happiness when they receive our custom-made catalog, i.e., how similar are their actual preferences to the ones that we predicted based on treating them as a part of a group.

2.2.1 External measure

We propose the following external measure for comparing different clustering solutions,

$$\Delta_{rank} = \frac{1}{K} \sum_{k=1}^{K} \frac{\sum_{n \in S_k} d_\tau(\pi_n, \rho_k)}{\mid S_k \mid}, \tag{2}$$

where ρ_k is the k-th cluster central label ranking. If Δ_{rank} is low it means that the clusters are pure in terms of label rankings, i.e., that the Kendall tau distances within each cluster are low.

2.2.2 Internal measures

One of the most common internal measures used is the within-cluster sum of squares (WCSS) (Han and Kamber 2001),

$$d_{feat} = \frac{1}{K} \sum_{k=1}^{K} \sum_{n \in S_k} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2, \tag{3}$$

where $\boldsymbol{\mu}_k$ is the mean of points in S_k. Note that K-means algorithm directly minimizes this measure. The main disadvantage of this measure is that it assumes globular clusters that have well defined centers. In contrast, in many real life applications we can encounter to non-globular clusters, such as chainlike and banana clusters, that would have high WCSS.

Due to limitations of WCSS, other measures have been proposed that measure how well the data point is matched to its cluster or the immediate neighboring points in its cluster, such as the Silhouette coefficient (Kaufmann and Rousseeuw 1990) or recently proposed average k-neighborhood distance measure d_{knh} (Kremer et al. 2011). The d_{knh} is obtained by simply calculating the average distance of each example $\mathbf{x}_n$ from cluster S_m to its k neighbors in S_m,

$$d_{knh}(n, S_m) = \frac{1}{k} \sum_{i \in knh(n, S_m)} \|\mathbf{x}_n - \mathbf{x}_i\|^2, \tag{4}$$

followed by calculating the average distance for a cluster S_m as $d_{knh}(S_m) = \frac{1}{|S_m|} \sum_{n \in S_m} d_{knh}(n, S_m)$, and averaging over all K clusters

$$d_{knh} = \frac{1}{K} \sum_{k=1}^{K} \frac{\sum_{n \in S_k} d_{knh}(n, S_m)}{|S_k|}. \tag{5}$$

One of the main conclusions in Kremer et al. (2011) was that the choice of k only has marginal influence on d_{knh} effectiveness. Following this conclusion, we used $k = 3$ in our evaluation.

2.2.3 Prediction measure

In applications such as target marketing, it is important to measure the quality of prediction on new examples for which we do not have customer preferences. Assuming, new examples $\mathbf{x}_n$ are assigned to one of the K cluster based on their features, we can measure the cluster prediction performance using the label ranking loss, defined as

$$loss_{LR} = \frac{1}{N} \sum_{n=1}^{N} \frac{2 \cdot d_\tau(\pi_n, \hat{\pi}_n)}{L \cdot (L-1)}, \tag{6}$$

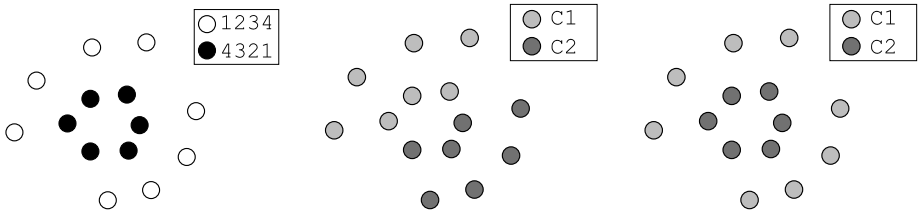
where π_n and $\hat{\pi}_n = \rho_m$ are true and predicted rankings for n-th example, respectively. The predicted ranking is the central ranking of the assigned cluster, where the assignment has been done using cluster abstractions $\boldsymbol{\mu}_k$, e.g., based on distances from cluster means $\boldsymbol{\mu}_k$. Note that $loss_{LR}$ can be used to calculate the Kendall's tau coefficient

$$c_\tau = 1 - 2 \cdot loss_{LR}. \tag{7}$$

Directly minimizing the Kendall tau loss (6) is hard, since it is not a smooth function. For this reason in the proposed clustering algorithms we assumed, and the experiments confirmed it, that likelihood maximization is an appropriate proxy criterion. Similar directions were taken in several other label ranking algorithms (Cheng et al. 2009, 2010).

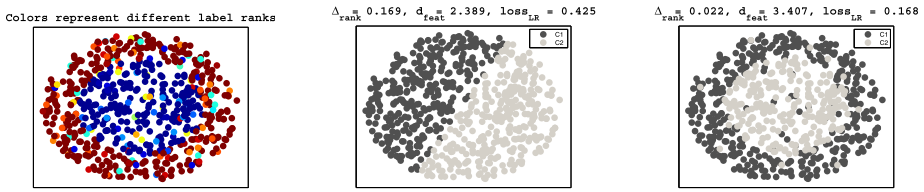
2.2.4 Discussion

To give insight into the performance measures, let us illustrate and analyze them on two different types of toy data. We will show that in some cases clustering solution with the



(a) Label rankings (b) Feature clustering (low d_{feat} high Δ_{rank}) (c) Label ranking clustering (high d_{feat} low Δ_{rank})

Fig. 2 Clustering of label ranking data toy example 1



(a) Label rankings (b) Feature clustering (c) Label ranking clustering

Fig. 3 Clustering of label ranking data toy example 1—simulation

lower Δ_{rank} leads to better predictive performance ($loss_{LR}$), while in other cases clustering solution with lower d_{feat} performs better in terms of $loss_{LR}$.

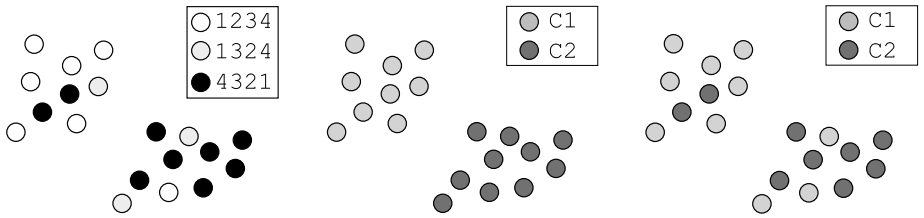
The first data type we investigated is shown in Fig. 2a. In Fig. 2b we show a potential clustering solution which clustered the data based on features only, without taking label rankings into account. This clustering solution has low d_{feat} and high Δ_{rank}. The new observations can be assigned to clusters based on distances from cluster means $\boldsymbol{\mu}_k$.

On the other hand, in Fig. 2c we show a potential clustering solution which clustered the data based on label rankings only, without taking features into account. The new observations can then be assigned to clusters based on 1-nearest neighbor rule.[1] This clustering solution has high d_{feat} and low Δ_{rank}, since we do not cluster in the feature space.

To further analyze different clustering solutions, we generated data of the described type (Fig. 3a) and performed actual feature clustering (using K-means) and label ranking clustering (using K-o'means (Kamishima and Akaho 2009)). We evaluated results in terms of d_{feat} and Δ_{rank} on original data and $loss_{LR}$ on test data, generated in a same way as the original data. By observing results in Fig. 3b and Fig. 3c and considering the ground truth we can conclude that in this particular case clustering solution with lower Δ_{rank} resulted in lower $loss_{LR}$.

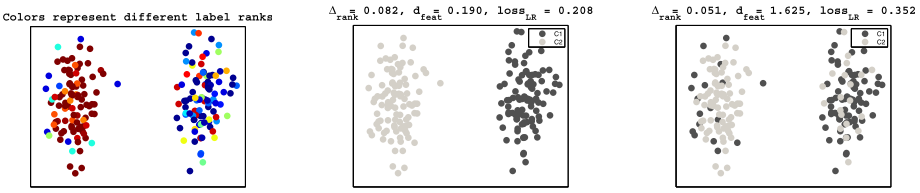
Further, let us consider a different type of data, as depicted in Fig. 4. Once again, clustering in feature space would result in low d_{feat} and high Δ_{rank}, while clustering in label ranking space would lead to high d_{feat} and low Δ_{rank}. Interestingly, this time the outcome of testing on simulated data is different (Fig. 5), as the clustering solution with lower d_{feat} leads to better performance in terms of $loss_{LR}$.

[1] In this case the cluster feature space representation $\boldsymbol{\mu}_k$ consists of all of its points, instead of the cluster mean. Using cluster means would lead to poor performance.



(a) Label rankings (b) Feature clustering (low d_{feat} high Δ_{rank}) (c) Label ranking clustering (high d_{feat} low Δ_{rank})

Fig. 4 Clustering of label ranking data toy example 2



(a) Label rankings (b) Feature clustering (c) Label ranking clustering

Fig. 5 Clustering of label ranking data toy example 2—simulation

We can conclude that low d_{feat} and/or low Δ_{rank} on their own are not necessarily the best indicators of the predictive performance in terms of $loss_{LR}$. However, they can give us valuable insights into the quality of clustering in both feature and output spaces, which is of great interest to, e.g., retailers in customer segmentation applications. As such we report all measures in our experiments. We have found that an algorithm with reasonable trade-off between Δ_{rank} and d_{feat} does best in practice.

3 Heuristic baselines for Label Ranking Clustering

We propose several baseline strategies for solving the Label Ranking Clustering problem. Some of the baselines models and the models described in the following sections use the probabilistic models for permutations such as the Mallows model (Mallows 1957) and the Plackett-Luce model (Plackett 1975; Luce 1959) to calculate the cluster central rankings given the rankings of the cluster instances. Therefore, in this section we first describe these ranking models, followed by description of the clustering baseline methods.

3.1 Ranking models

The Mallows model is a distance-based probabilistic model for permutations. The probability of any permutation ρ of L labels is given in terms of a permutation distance d as follows

$$\mathbb{P}(\rho \mid \theta, \pi) = \frac{\exp(-\theta\, d(\pi, \rho))}{Z(\theta, \rho)}, \tag{8}$$

where $\rho \in T$ is candidate permutation, π is central ranking, $\theta \in \mathbb{R}$ is dispersion parameter, $Z(\theta, \pi) = \sum_{\rho \in T} \exp(-\theta\, d(\rho, \pi))$ is a normalization constant, and d is, in our case, the Kendall's tau distance d_τ. The maximum probability is assigned to the central ranking π.

Given N independent label rankings $\{\pi_n\}, n = 1, \ldots, N$, the Mallows model can find a central ranking using the maximum likelihood principle. Let us discuss how it can be used to derive cluster centroids $\{\rho_k\}, k = 1, \ldots, K$ for K clusters, in general. Given $|S_k|$ label rankings $\{\pi_n\}, \mathbf{x}_n \in S_k$ for cluster k and assuming independence, the probability that we observe $\boldsymbol{\pi}_k = \{\pi_n\}, \mathbf{x}_n \in S_k$ is

$$\mathbb{P}(\boldsymbol{\pi}_k \mid \theta_k, \rho_k) = \prod_{n \in S_k} \mathbb{P}(\pi_n \mid \theta_k, \rho_k). \tag{9}$$

The maximum likelihood estimate (MLE) of k-th cluster central ranking ρ_k is the one that minimizes

$$loss_{LR}(k) = \frac{1}{|S_k|} \sum_{n \in S_k} \frac{2 \cdot d_\tau(\pi_n, \rho_k)}{L \cdot (L-1)}, \tag{10}$$

where the solution can be found by exhaustive search. The MLE of θ_k is found from the mean observed distance from ρ_k, $\frac{1}{|S_k|} \sum_{n \in S_k} d_\tau(\pi_n, \rho_k)$ by line search.

The disadvantages of the Mallows model are that it has high computational complexity of $\mathcal{O}(L!)$, and that it cannot directly model incomplete label ranks. However, there are several papers (Cheng et al. 2009; Dwork et al. 2001; Lu and Boutilier 2011) that address these issues by introducing fast approximations which scale linearly with L, work well in practice, and can deal with partial rankings (Cheng et al. 2009) and pairwise preferences (Lu and Boutilier 2011).

To mitigate the first issue we make use of the approximate solution (Cheng et al. 2009) that uses a simple Borda count algorithm (de Borda 1781). Having $|S_k|$ label rankings $\{\pi_n\}, \mathbf{x}_n \in S_k$, the central rank is found by voting. Each π_n votes in the following manner. The label which is ranked first in π_n gets L votes, the second-ranked receives $L-1$ votes, etc. Finally, all the votes from $\pi_n, \mathbf{x}_n \in S_k$ are summed up and the label with the most votes is ranked first in ρ_k, the label with the second most votes is ranked second in ρ_k, etc. The approximation is valid as it has been shown that Kendall's tau is well approximated by Spearman's rank correlation (Coppersmith et al. 2006), whose median can be computed using Borda.

To solve the second issue Borda count is modified (Cheng et al. 2009) such that partial rankings π_n of $L_n < L$ labels vote in the following manner. The label ranked j-th ($j \leq L_n$) receives $(L_n - j + 1) \cdot (L+1)/(L_n+1)$ votes, while all the missing labels receive $(L+1)/2$ votes. Once ρ_k is obtained, for each incomplete ranking π_n in cluster S_k, the most probable extension to full rank π_n^* is found such that $loss_{LR}(\rho_k, \pi_n^*)$ is minimized. Finally, original Borda count is used to find ρ_k from $\{\pi_n^*\}, \mathbf{x}_n \in S_k$. This procedure iterates until ρ_k converges.

The Plackett-Luce model is an extension of the Bradley-Terry model (Bradley and Terry 1952) for comparisons involving three or more alternatives. It is defined by the score vector $\mathbf{v} = (\mathbf{v}(1), \mathbf{v}(2), \ldots, \mathbf{v}(L)) \in \mathbb{R}_+^L$. The probability of any ranking of L labels is expressed in terms of $\mathbf{v}$ as

$$\mathbb{P}(\pi \mid \mathbf{v}) = \prod_{i=1}^{L} \frac{\mathbf{v}(\pi(i))}{\sum_{j=i}^{L} \mathbf{v}(\pi(j))}. \tag{11}$$

If the score vector $\mathbf{v}$ is known, the ranking with the highest posterior probability $\pi^* = \arg\max_\pi(\mathbb{P}(\pi|\mathbf{v}))$ can be found by simply sorting the labels in the descending order of the

corresponding scores. Let us consider a simple case with 3 labels $\{a, b, c\}$. The probability of a specific ranking of labels (e.g., $\pi = (c, a, b)$) can be expressed as

$$\mathbb{P}\big(\pi = (c, a, b)|\mathbf{v}\big) = \frac{\mathbf{v}(c)}{\mathbf{v}(c) + \mathbf{v}(a) + \mathbf{v}(b)} \cdot \frac{\mathbf{v}(a)}{\mathbf{v}(a) + \mathbf{v}(b)} \cdot \frac{\mathbf{v}(b)}{\mathbf{v}(b)}, \tag{12}$$

where the first multiplier presents the probability $\mathbb{P}(c$ is ranked 1st), the second presents the probability $\mathbb{P}(a$ is ranked 2nd $\mid c$ is ranked 1st) and the third presents the probability $\mathbb{P}(b$ is ranked 3rd $\mid c$ is ranked 1st $\wedge$ a is ranked 2nd). For example, assuming the label scores $v(a) = 3$, $v(b) = 1$ and $v(c) = 5$, and plugging the scores back to (12) we get $\mathbb{P}(\pi = (c, a, b)|\mathbf{v}) = 0.417$.

One of the properties of the PL distribution is that it is internally consistent. As shown in Guiver and Snelson (2009), given two sets of labels $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{B} \subset \mathcal{A}$, the probability of a particular ordering of labels in $\mathcal{B}$, marginalized over all possible unknown positions of the labels in $\mathcal{A} \setminus \mathcal{B}$, is exactly the same as the Plackett-Luce probability of ordering labels from $\mathcal{B}$ independently from $\mathcal{A}$. Thus, the probability of a particular ordering of labels does not depend on the subset from which the labels are assumed to be drawn (Hunter 2004). This gives the Plackett-Luce model the ability to model partial rankings of only some of the labels, or rankings of the top few labels. The probability of partial ranking with only some of the labels, $y_{\pi(1)} \succ_{\mathbf{x}} y_{\pi(2)} \succ_{\mathbf{x}}, \ldots, \succ_{\mathbf{x}} y_{\pi(L_t)}$, is given by an expression of exactly the same form as (11), except that the number of elements is L_t instead of L, where $L_t \leq L$,

$$\mathbb{P}(\pi \mid \mathbf{v}) = \prod_{i=1}^{L_t} \frac{\mathbf{v}(\pi(i))}{\sum_{j=i}^{L_t} \mathbf{v}(\pi(j))}. \tag{13}$$

The disadvantage of the PL model is that it cannot learn from arbitrary pair-wise preferences that cannot form a permutation (e.g., $1 \succ 5, 2 \succ 1, 3 \succ 4$). One solution worth investigating could be to split the instance with such preferences into several instances that form a permutation, i.e., one with ranking $2 \succ 1 \succ 5$ and another with $3 \succ 4$, where instances generated in such way would share the same features.[2] By modifying data in this way, the PL model could be directly applied.

Given N independent label ranking observations $\{\pi_n\}, n = 1, \ldots, N$, the Plackett-Luce model can find a central ranking using the maximum likelihood principle (Cheng et al. 2010). Let us discuss how it can be used to derive cluster centroids $\{\rho_k\}, k = 1, \ldots, K$. Given $|S_k|$ label rankings $\{\pi_n\}$ associated with $|S_k|$ feature vectors $\mathbf{x}_n \in S_k$, and assuming independence between observations, the probability that we observe $\boldsymbol{\pi}_k = \{\pi_n\}, \mathbf{x}_n \in S_k$ can be written as

$$\mathbb{P}(\boldsymbol{\pi}_k \mid \mathbf{v}_k) = \prod_{n \in S_k} \prod_{i=1}^{L_n} \frac{\mathbf{v}_k(\pi(i))}{\sum_{j=i}^{L_n} \mathbf{v}_k(\pi(j))}. \tag{14}$$

The k-th cluster PL vector $\mathbf{v}_k$, and thus the central ranking ρ_k, can be found by maximum likelihood principle, where the optimal $\mathbf{v}_k$ is the one which maximizes the log-likelihood function

$$l(\mathbf{v}_k) = \sum_{n \in S_k} \sum_{i=1}^{L_n} \left(\log \mathbf{v}_k\big(\pi(i)\big) - \log \sum_{j=i}^{L_n} \mathbf{v}_k\big(\pi(j)\big) \right). \tag{15}$$

[2] We thank one of the reviewers for pointing out this idea.

3.2 Baselines

We propose three types of baseline approaches for label ranking clustering. In addition we will measure how accurately the data can be represented using a single cluster. We will refer to this method as 1-Rank.

1-**Rank** serves to show how appropriate it would be to consider all examples as a single cluster, $K = 1$. Intuitively, this would correspond to designing a single catalog for all customers. In this approach we use the Mallows model to derive a single central ranking ρ_K using entire data. Note that it is straightforward to use PL model for this purpose as well.

3.2.1 Feature clustering → central rank

K-**means → central rank** algorithm first performs K-means clustering based on the instance features only, without taking their label rankings into account. This forms cluster representation in feature space in terms of cluster centroids $\boldsymbol{\mu}_k$. It then derives central rankings ρ_k for each cluster from label rankings that belong to this cluster using the Mallows model ($\mathbf{Km}_{Mal}$) or the Plackett-Luce model ($\mathbf{Km}_{Pl}$). This approach is expected to work well when clusters in the feature space correspond to clusters in the label space.

3.2.2 Rank clustering → feature rule

Rank clustering → FR first clusters the label rankings using the unsupervised label ranking approach to obtain K clusters in label ranking space, by assigning each instance $\mathbf{x}_n$ to one central rankings $\rho_k, k = 1, \ldots, K$. After this, it remains to form a feature rule (FR), i.e., an abstraction of the formed clusters in the feature space $\boldsymbol{\mu}_k$ that will be used and to assign new instances to clusters based on features only. For example, $\boldsymbol{\mu}_k$ could be formed by storing all the points for that cluster and using a 1-nearest neighbor rule for assignments, or $\boldsymbol{\mu}$ could be a decision tree that splits the feature space based on the obtained central rankings.

We consider three rank clustering methods for the first stage of the described algorithm:

(1) **Naïve** first sorts all available permutations of L labels by the number of times they appear in the data set. The top K label ranks determine cluster central rankings and define the K classes. The remaining label ranks are assigned the class of the closest central ranking, with respect to the normalized Kendall tau distance. If incomplete label rankings are present, we first determine their extensions by finding the most probable positions of the missing labels using a modified Borda count algorithm on the instance's neighborhood of size $K_{mal} = 20$. This approach is expected to work well when the clusters are evenly distributed in the label space and top K ranks really correspond to cluster central rankings. However, if the clusters are unbalanced, it is not likely that this simple procedure will discover the underlying label space centroids.

(2) **EBMS** is a permutation clustering technique from Meilă and Bao (2010). Based on ranking distribution in the data set and their similarities, training permutations are divided into K clusters, where each cluster is represented with a central ranking. Note that the EBMS algorithm is able to cluster incomplete rankings.

(3) K-**o'means** (Kamishima and Akaho 2009). Data set is randomly split into K clusters, then for each cluster the central ranking is found using Borda count on the label ranks. Data points are then reassigned to clusters such that Kendall's tau distance between data point's label ranking and cluster's central ranking is minimized. The procedure repeats until there are no more reassignments or maximum number of iterations is reached. Incomplete rankings are tackled using modified Borda count.

For the second stage of the described algorithm, we use a Decision Tree algorithm. After the K central rankings are found, we grow a decision tree with these rankings at leafs. Beginning with the entire data set T, we choose the split into T^+ and T^- by trying out each of the K candidate ranks for ρ_{T^+} and ρ_{T^-}, and decide on the split that minimizes overall loss (18). This gave rise to three methods, namely $\textbf{Naïve}_{RT}$, $\textbf{ebms}_{RT}$ and $K\textbf{-o'm}_{RT}$.

We also experimented with multi-class kernel SVM with K classes instead of the Decision Trees in the second stage. This approach performed well in some cases but poorly in others, due to over-fitting based on the assigned classes. Moreover, the resulting clusters were not compact in the feature space due to use of kernel. Mostly because of the second issue we abandoned the SVM approach in this stage.

3.2.3 Naïve K-means

In this baseline approach, label rankings π are treated as one of the features. This is done by adding L additional attributes to the feature vector $\mathbf{x}$, that results in the new vector of length $d + L$. Value of $(d + j)$-th attribute is set to jth label position in the ranking π, or $L/2$ if the particular label is not available in π. After this preprocessing stage, the newly formed data is clustered using benchmark K-means algorithm. Central rankings ρ_k for each cluster are derived from obtained cluster centroids by sorting the last L attributes in the ascending order. These rankings are used to predict the label ranking of the new instance when its cluster membership is determined by finding the nearest cluster centroid in the original feature space.

4 RankTree label ranking clustering algorithm

Decision Trees (Breiman et al. 1984) are one of the most popular classification and regression algorithms, extensively studied and used due to their good performance, intuitive nature and desirable training times.

Several recent papers (Cheng et al. 2009; Yu et al. 2010) proposed to use Decision Trees with binary splits for solving Label Ranking problems. For example, in Cheng et al. (2009), the authors propose to recursively grow a decision tree by finding optimal splits that make the subsets as pure as possible in terms of label rankings at each stage. The resulting leaf nodes are associated with label rankings which can potentially be incomplete (if not all labels are present in the partitioned subset of data).

Formally, given current iterate data subsample T, it is split into two subsets T^+ and T^- by fitting a Mallows model to each subset and measuring the variance of ranks in each subset. The results are the subset central rankings ρ^+ and ρ^- and the spread parameters θ^+ and θ^- that estimate the variance. The optimal split is the one that maximizes

$$|T|^{-1}\big(\big|T^+\big|\theta_{T^+} + \big|T^-\big|\theta_{T^-}\big). \tag{16}$$

Comparing our method to the RankTree algorithm from Cheng et al. (2009) was not directly possible, as the algorithm is not limited to K possible outputs. We therefore proposed two ways of modifying the algorithm such that it can perform the supervised clustering task. The first modification includes pruning the constructed tree until it has distinct K ranking in leafs. The second modification we propose as a new algorithm includes iteratively setting K possible rankings in advance and growing the tree until loss measure stops improving.

The first modification can be achieved by growing the tree until the stopping criteria is met, followed by pruning until the pruning criteria is met. The stopping criteria is satisfied

when all the instances in T have the same label ranks or the node becomes very small (e.g., it contains less than 5 instances). The pruning criteria is met either when there are no more than K different label rankings in the tree leafs, or the number of leafs is K. We refer to this RankTree algorithm for supervised clustering as $\mathbf{RT}_{prune}$.

The tree is grown using the same procedure as in Cheng et al. (2009), followed by pruning the tree using same data used for growing. This is done by greedy removal of splits which contribute to accuracy the least. We keep track of the rank losses at each node T and remove the leafs of parent node with minimum value of

$$\sum_{n\in T} d_\tau(\pi_n, \rho_T) - \left(\sum_{n\in T^+} d_\tau(\pi_n, \rho_{T^+}) + \sum_{n\in T^-} d_\tau(\pi_n, \rho_{T^+})\right). \tag{17}$$

Note that it is possible for ρ_{T^+} and ρ_{T^-} to be partial ranks. In this case, these rankings are expended to full ranks based on the predecessor label rank as proposed by Cheng et al. (2009). The disadvantage of this strategy is that the final tree will have small number of splits (in the worst case $K-1$), which may not be enough to adequately cluster the feature space.

For this reason we propose the second type of RankTree algorithm, which can use arbitrary number of splits. It uses an idea of iterative encoder and decoder design in General Source Coding (Lam and Reibman 1993; Megalooikonomou and Yesha 2000). The algorithm starts by randomly selecting K ranks and growing a tree limited to assigning one of the K ranks to the leafs. Data set T is iteratively split into two data subsets T^+ and T^- by trying each of the K rankings for the subset central rankings ρ^+ and ρ^- and selecting the split which minimizes

$$\sum_{n\in T^+} d_\tau(\pi_n, \rho_{T^+}) + \sum_{n\in T^-} d_\tau(\pi_n, \rho_{T^-}). \tag{18}$$

Once the tree is grown, the K ranks are recalculated by finding a central ranking of examples assigned to each of the K clusters using the modified Borda count (Cheng et al. 2009). The procedure then continues to growing a new tree, etc. The iterative procedure is terminated when the training set $loss_{LR}$ converges, e.g., stops decreasing by more than 10^{-4}. Optional pruning step can be performed after every iteration if a validation data set is available, which is separate from training data. Starting at the leafs, a split at node a is removed if the result of (17) on pruning data is negative. The pseudo code is given in Algorithm 1. We will refer to this algorithm as $\mathbf{RT}_{iter}$.

Algorithm 1 $\mathbf{RT}_{iter}$ algorithm for Supervised Clustering of Label Ranking Data

Input: train data $D = (\mathbf{x}_n, \pi_n), n = 1, \ldots, N$, valid. (optional) $D_e = (\mathbf{x}'_n, \pi'_n), n = 1, \ldots, N_e$, K
Output: K clusters $S_k = (\boldsymbol{\mu}, \rho_k), k = 1, \ldots, K$, where $\boldsymbol{\mu}$ is the Decision Tree
1. **Initialize** $\rho_k, k = 1, \ldots, K$ to random total orders of L labels
2. **while** ($loss_{LR}$ decreases by more than 10^{-4})
3. **Grow** decision tree $\boldsymbol{\mu}$ recursively starting from $T = D$
   3.1. **For** every possible split across each feature dimension of T
   3.2. **Try** each $\rho_k, k = 1, \ldots, K$ for ρ_{T^+} and ρ_{T^-}
   3.3. **Split** in the place minimizing $\sum_{n\in T^+} d_\tau(\pi_n, \rho_{T^+}) + \sum_{n\in T^-} d_\tau(\pi_n, \rho_{T^-})$
   3.4. **Stop** when $\rho_{T^+} = \rho_{T^-}$ or $|T| \leq 5$
5. **Prune** RankTree $\boldsymbol{\mu}$ using D_e (optional)
6. **Recalculate** $\rho_k, k = 1, \ldots, K$, by finding central rankings of $\{\pi_n, n \in S_k\}$ (Mallows)
7. **Calculate** $loss_{LR}$

5 The Plackett-Luce Mixture Model

In this section, we describe the details involving the proposed Plackett-Luce Mixture Model for supervised clustering of label ranking data.

5.1 Model specification

The Plackett-Luce Mixture Model (MM-PL) has the following setup. The feature space is divided into K clusters, where each cluster $S_k = (\boldsymbol{\mu}_k, \rho_k), k = 1, \ldots, K$, is represented as a union of Voronoi cells $\boldsymbol{\mu}_k$ defined by a set of prototypes $\boldsymbol{\mu}_k = \{\mathbf{m}_p, p = 1, \ldots, P\}$, where $\mathbf{m}_p$ is a d-dimensional vector in input space. In the label ranking space each cluster is represented by a cluster central ranking ρ_k, defined as a Plackett-Luce label score vector $\mathbf{v}_k$. The model is based on the assumption that the feature space can be well decomposed using a Voronoi diagram.

The resulting model is completely defined by $K \cdot P$ prototypes and K Plackett-Luce score vectors, $\{(\{\mathbf{m}_p\}_k, \mathbf{v}_k), k = 1, \ldots, K, p = 1, \ldots, P\}$. The optimal prototype positions and label score vectors, with respect to a desired loss function, are learned in a supervised manner using a set of N independent observations $D = \{(\mathbf{x}_n, \pi_n), n = 1, \ldots, N\}$.

The starting point in the algorithm design is to introduce the probability $\mathbb{P}(p \mid \mathbf{x}_n)$ of assigning observation $\mathbf{x}_n$ to the p-th prototype that is dependent on their (Euclidean) distance in the feature space. Let us assume that the probability density $\mathbb{P}(\mathbf{x}_n)$ is described by a mixture model

$$\mathbb{P}(\mathbf{x}_n) = \sum_{p=1}^{P \cdot K} \mathbb{P}(\mathbf{x}_n \mid p) \cdot \mathbb{P}(p), \tag{19}$$

where $P \cdot K$ is the total number of prototypes, $\mathbb{P}(p)$ is the prior probability that a data point is generated by a particular prototype, and $\mathbb{P}(\mathbf{x}_n \mid p)$ is the conditional probability that p-th prototype generates data point $\mathbf{x}_n$.

Let us represent the conditional density function $\mathbb{P}(\mathbf{x}_n \mid p)$ with the normalized exponential form $\mathbb{P}(\mathbf{x}_n \mid p) = \theta(p) \cdot \exp(f(\mathbf{x}_n, \mathbf{m}_p))$ and consider a Gaussian mixture with $\theta(p) = (2\pi\sigma_p^2)^{-1/2}$ and $f(\mathbf{x}_n, \mathbf{m}_p) = -\|\mathbf{x}_n - \mathbf{m}_p\|^2/2\sigma_p^2$. We assume that all prototypes have the same standard deviation (width) σ_p and the same prior, $\mathbb{P}(p) = 1/(P \cdot K)$. Given this, using the Bayes' rule, we can write the assignment probability as

$$w_{np} \equiv \mathbb{P}(p \mid \mathbf{x}_n) = \frac{\exp\left(-\|\mathbf{x}_n - \mathbf{m}_p\|^2/2\sigma_p^2\right)}{\sum_{v=1}^{P \cdot K} \exp\left(-\|\mathbf{x}_n - \mathbf{m}_v\|^2/2\sigma_p^2\right)}. \tag{20}$$

Finally, to develop the cost function for label ranking clustering framework we consider a probabilistic assignment $\mathbb{P}(k \mid \mathbf{x}_n)$ defined as the probability of assigning data point $\mathbf{x}_n$ to the k-th cluster,

$$g_{nk} \equiv \mathbb{P}(k \mid \mathbf{x}_n) = \frac{\sum_{p \in S_k} \exp\left(-\|\mathbf{x}_n - \mathbf{m}_p\|^2/2\sigma_p^2\right)}{\sum_{v=1}^{P \cdot K} \exp\left(-\|\mathbf{x}_n - \mathbf{m}_v\|^2/2\sigma_p^2\right)}. \tag{21}$$

For the compactness of notation, we defined $w_{np} \equiv \mathbb{P}(p \mid \mathbf{x}_n)$ and $g_{nk} \equiv \mathbb{P}(k \mid \mathbf{x}_n)$. We propose the following mixture model for the posterior probability $\mathbb{P}(\boldsymbol{\pi} \mid \mathbf{x}_n)$.

Assuming that the clusters are mutually exclusive, i.e., that the probability of two different clusters generating an instance ranking π given the input $\mathbf{x}_n$ is zero, we can write the

overall probability as

$$\mathbb{P}(\boldsymbol{\pi} \mid \mathbf{x}_n) = \sum_{k=1}^{K} \mathbb{P}(k \mid \mathbf{x}_n) \cdot \mathbb{P}(\boldsymbol{\pi} \mid k, \mathbf{x}_n), \tag{22}$$

where $\mathbb{P}(\boldsymbol{\pi} \mid k, \mathbf{x}_n)$ is the probability of ranking $\boldsymbol{\pi}$ if $\mathbf{x}_n$ is generated by k-th cluster. In our approach, we assume the ranking $\boldsymbol{\pi}$ corresponds to the PL model, and express $\mathbb{P}(\boldsymbol{\pi} \mid k, \mathbf{x}_n)$ as $\mathbb{P}(\boldsymbol{\pi} \mid \mathbf{v}_k, \mathbf{x}_n)$ defined in (11).

$$e_{nk} \equiv \mathbb{P}(\boldsymbol{\pi} \mid k, \mathbf{x}_n) = \mathbb{P}(\boldsymbol{\pi} \mid \mathbf{v}_k, \mathbf{x}_n) = \prod_{i=1}^{L_n} \frac{\mathbf{v}_k(\pi_n(i))}{\sum_{j=i}^{L_n} \mathbf{v}_k(\pi_n(j))}. \tag{23}$$

Based on the model in (11), example $\mathbf{x}_n$ is assigned to the clusters probabilistically and its label ranking probability is a weighted average of ranking probabilities assigned to the clusters. Similar competition between the experts in the regression setting is modeled in Weigend et al. (1995). Note that the mixture model assumes the conditional independence between $\mathbf{x}_n$ and $\boldsymbol{\pi}$ given k, $\mathbb{P}(\boldsymbol{\pi} \mid k, \mathbf{x}_n) = \mathbb{P}(\boldsymbol{\pi} \mid k)$.

5.2 Model learning

Given the training data set D and assuming the mixture model (22), we can find the optimal model parameters $\lambda = \{\{\mathbf{m}_p\}_k, \mathbf{v}_k, k = 1, \ldots, K, \sigma_p\}$ as the ones that maximize the likelihood,

$$\mathbb{P}(\boldsymbol{\pi} \mid \lambda) = \prod_{n=1}^{N} \sum_{k=1}^{K} \mathbb{P}(k \mid \mathbf{x}_n) \cdot \prod_{i=1}^{L_n} \frac{\mathbf{v}_k(\pi_n(i))}{\sum_{j=i}^{L_n} \mathbf{v}_k(\pi_n(j))} = \prod_{n=1}^{N} \sum_{k=1}^{K} g_{nk} e_{nk}, \tag{24}$$

where L_n is the number of labels in, potentially incomplete, label ranking π_n for n-th training instance. The maximum likelihood estimation of λ can be found by minimizing the negative log-likelihood function.

$$l(\lambda) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \mathbb{P}(k \mid \mathbf{x}) \cdot \prod_{i=1}^{L_n} \frac{\mathbf{v}_k(\pi_n(i))}{\sum_{j=i}^{L_n} \mathbf{v}_k(\pi_n(j))}. \tag{25}$$

In principle, it should be possible to find a local minimum of non-convex function with standard gradient descent. However, for mixtures such as the one we defined, it is hard to learn at the same time both individual mixture parameters and splits of the input space using gradient descent (Weigend et al. 1995). Moreover, the sum inside the logarithm in (25) makes the task even more challenging.

For this reason we resort to a more natural alternative to minimization of $l(\lambda)$, the Expectation-Maximization (EM) algorithm (Dempster et al. 1977). To reformulate the problem such that the Expectation-Maximization approach can be applied, we need to adopt the assumption that some variables are "hidden". Similarly to the approach in Weigend et al. (1995), we treat the cluster membership as a hidden variable. These hidden indicator variables I_{nk} are defined as

$$I_{nk} = \begin{cases} 1, & \text{if } \mathbf{x}_n \text{ is generated by } k\text{-th cluster,} \\ 0, & \text{otherwise.} \end{cases} \tag{26}$$

Note that this assumption of one cluster being responsible for each instance, is identical to the assumption that allowed us to write (22). We can now rewrite (25) as

$$l(\lambda) = -\ln \prod_{n=1}^{N} \prod_{k=1}^{K} \left[\mathbb{P}(k \mid \mathbf{x}) \cdot \prod_{i=1}^{L_n} \frac{\mathbf{v}_k(\pi_n(i))}{\sum_{j=i}^{L_n} \mathbf{v}_k(\pi_n(j))} \right]^{I_{nk}}. \tag{27}$$

Further, by taking the logarithm, we end up with a simple double-sum, yielding a more tractable log-likelihood function,

$$l(\lambda) = -\sum_{n=1}^{N} \sum_{k=1}^{K} I_{nk} \ln \left[\mathbb{P}(k \mid \mathbf{x}) \cdot \prod_{i=1}^{L_n} \frac{\mathbf{v}_k(\pi_n(i))}{\sum_{j=i}^{L_n} \mathbf{v}_k(\pi_n(j))} \right]. \tag{28}$$

The objective function (28) can now be optimized by iterating between E- and M-steps of the EM algorithm. In the E-step, the expected values for I_{nk}, denoted as h_{nk}, are estimated assuming that the model parameters λ are known,

E-step:

$$\begin{aligned} h_{nk} &= E[I_{nk} \mid \mathbf{x}_n, \pi_n, \lambda] = \mathbb{P}(k \mid \mathbf{x}_n, \pi_n) = \frac{\mathbb{P}(k, \pi_n \mid \mathbf{x}_n)}{\mathbb{P}(\pi_n \mid \mathbf{x}_n)} \\ &= \frac{g_{nk} \cdot \mathbb{P}(\pi_n \mid \mathbf{x}_n, k)}{\mathbb{P}(\pi_n \mid \mathbf{x}_n)} = \frac{g_{nk} \cdot \mathbb{P}(\pi_n \mid \mathbf{x}_n, k)}{\sum_{v=1}^{K} g_{nv} \cdot \mathbb{P}(\pi_n \mid \mathbf{x}_n, v)} = \frac{g_{nk} e_{nk}}{\sum_{v=1}^{K} g_{nv} e_{nv}}. \end{aligned} \tag{29}$$

In the M-step, the parameters of the model are updated assuming the I_{nk} values are known, and replacing each I_{nk} in (28) by its expected value h_{nk}. Note that there exists a strict requirement that $\mathbf{v}_k \in \mathbb{R}_+^L, k = 1, \ldots, K$ and thus we have to guarantee that all elements of $\mathbf{v}$ are positive. In this setting, learning becomes a constrained optimization problem. Since gradient ascent cannot be directly applied to a constrained optimization problem, we propose two alternatives for transforming the original constrained optimization problem to a new optimization problem, which is unconstrained, and thus simpler to solve.

In the first approach we minimize the negative log-likelihood $l(\lambda)$ with respect to $\mathbf{z}_k = \log(\mathbf{v}_k)$ instead of $\mathbf{v}_k$. This converts the problem into the unconstrained optimization with respect to $\mathbf{z}_k$, which can now be solved using gradient ascent approach, since there are no strict requirements for $\mathbf{z}_k$. Once the learning phase is finished, $\mathbf{v}_k$ is obtained as $\mathbf{v}_k = \exp(\mathbf{z}_k)$. Similar technique was used in Qin et al. (2008). The summary of the resulting M-step ($v.1$) in the stochastic mode is given as follows,

M-step (v.1):

$$\begin{aligned} \mathbf{m}_p^{new} &= \mathbf{m}_p^{old} - \gamma \cdot w_{np} \frac{(1 - h_{nk})}{g_{nk}} \frac{(\mathbf{x}_n - \mathbf{m}_p)}{\sigma_p^2}, \quad \mathbf{m}_p \in S_k \\ \log\left(\mathbf{v}_k\left(\pi_n(c)\right)^{new}\right) &= \log\left(\mathbf{v}_k\left(\pi_n(c)\right)^{old}\right) - \gamma \cdot \mathbf{v}_k\left(\pi_n(c)\right) h_{nk} \\ &\quad \cdot \left(\frac{1}{\mathbf{v}_k(\pi_n(c))} - \sum_{i=1}^{c} \frac{1}{\sum_{j=i}^{L_n} \mathbf{v}_k(\pi_n(j))} \right), \end{aligned} \tag{30}$$

where γ is the learning rate, and $c \in \{1, \ldots, L_n\}$.

The second alternative is to resort to a closed form solution for $\mathbf{v}_k$ score vectors, derived using tools from the statistical literature. Minorization-Maximization (MM) algorithm proposed in Hunter (2004) has been shown to be particularly suitable for calculating maximum

likelihood estimates of the Plackett-Luce model. Incorporating the MM algorithm into our MM-PL results in a second version of the M-step ($v.2$) given as,

M-step (v.2):

$$\begin{aligned} \mathbf{m}_p^{new} &= \mathbf{m}_p^{old} - \gamma \cdot w_{np} \frac{(1-h_{nk})}{g_{nk}} \frac{(\mathbf{x}_n - \mathbf{m}_p)}{\sigma_p^2}, \quad \mathbf{m}_p \in S_k \\ \mathbf{v}_k\big(\pi_n(c)\big)^{new} &= \frac{\sum_{n=1}^{N} h_{nk} \cdot b(n,c)}{\sum_{n=1}^{N} h_{nk} \sum_{i=1}^{L_n-1} \delta(n,i,c)[\sum_{j=i}^{L_n} \mathbf{v}_k(\pi_n(j))]^{-1}}, \end{aligned} \tag{31}$$

where

$$b(n,c) = \begin{cases} 1, & \text{if } \pi_n^{-1}(c) < L_n, \\ 0, & \text{otherwise}, \end{cases} \tag{32}$$

and

$$\delta(n,i,c) = \begin{cases} 1, & \text{if } \pi_n^{-1}(c) \geq i, \\ 0, & \text{otherwise}. \end{cases} \tag{33}$$

The advantage of the M-step ($v.2$) is that the closed form solution for score vectors $\mathbf{v}_k$ directly ensures that elements of vector $\mathbf{v}_k$ are positive. It also avoids potential gradient descent issues such as initialization.

An important issue to be addressed is the choice of the parameter σ_p. Parameter σ_p controls the fuzziness of the distribution. For $\sigma_p = 0$ the assignments become deterministic, while for $\sigma_p \to \infty$ the assignments become uniform, regardless of the distance. One option is for σ_p to be treated as a parameter to be optimized such that $l(\lambda)$ is minimized. However, it is not necessarily the best approach. In this work, we are treating σ_p as an annealing parameter that is initially set to a large value, and is then decreased towards zero using scheme $\sigma_p(n+1) = \sigma_p(0) \cdot \sigma_T/(\sigma_T + n)$, where σ_T is the decay parameter. The purpose of annealing is to facilitate convergence towards a good local optimum of $l(\lambda)$. We note that this strategy has been used in soft prototype approaches by other researchers (Seo et al. 2003; Grbovic and Vucetic 2009).

Let us denote by p_{NN} the index of the nearest prototype for instance $\mathbf{x}_n$. As σ_p decreases towards zero, the assignment probability $\mathbb{P}(p_{NN}|\mathbf{x}_n)$ approaches one, and the assignment probabilities of the remaining prototypes approach zero. As a result, in the limit $p(\boldsymbol{\pi}|\mathbf{x}_n)$ from (22) becomes $\lim_{\sigma_p \to 0} \mathbb{P}(\boldsymbol{\pi}|\mathbf{x}_n) = \mathbb{P}(\boldsymbol{\pi}|k_{NN})$, where k_{NN} is the p_{NN}-th prototype's cluster. Therefore, the label rank for instance $\mathbf{x}_n$ can be predicted by using the label score vector of its nearest prototype's cluster. Since the optimal ranking for any cluster π_k, $k = 1, \ldots, K$ can be obtained by simply sorting the elements of label score vector $\mathbf{v}_k$, it could be performed only once at the end of the training procedure, and used in the prediction phase to speed-up the prediction. The pseudo-code for the v.1 M-step EM version of our algorithm is given in Algorithm 2.

6 Computational complexity

In this section we discuss the computational complexity of the considered supervised clustering algorithms listed in Table 1.

The first type of baseline algorithms, $\mathbf{Km}_{Mal}$ and $\mathbf{Km}_{Pl}$ take $\mathcal{O}(KN+NL)$ and $\mathcal{O}(KN+NL)$ time respectively. This is because they use the K-means algorithm in the first stage,

Algorithm 2 MM-PL algorithm for Supervised Clustering of Label Ranking Data

Inputs: train data $D = (\mathbf{x}_n, \pi_n), n = 1, \ldots, N$, K, P
Output: $\{\mathbf{m}_p\}_k, \mathbf{v}_k, k = 1, \ldots, K, p = 1, \ldots, P$ that define K clusters $S_k = (\boldsymbol{\mu}_k, \rho_k), k = 1, \ldots, K$
1. **Initialize** $\{\mathbf{m}_p\}_k$ to P random points from D, $\mathbf{v}_k$ by assigning $\epsilon \sim N(1, 0.1)$ to each label score, $\gamma_0 = 0.03$, σ_p to data D variance
2. **while** ($loss_{LR}$ decreases by more than 10^{-4})
3. **E-step** for each $(\mathbf{x}_n, \pi_n) \in D$ calculate h_{nk} using (29) for $k = 1, \ldots, K$
4. **M-step** for each $(\mathbf{x}_n, \pi_n) \in D$ update $\{\mathbf{m}_p\}_k$ and $\mathbf{v}_k$ using (30) for $p = 1, \ldots, P$
5. $k = 1, \ldots, K$, $\gamma(n) = \gamma_0 \cdot \gamma_T/(\gamma_T + n)$, $\sigma(n) = \sigma_0 \cdot \sigma_T/(\sigma_T + n)$, $\sigma_T = \gamma_T = 8N$
4. **Calculate** $loss_{LR}$
5. **Shuffle** data D
6. **Calculate** ρ_k by sorting $\mathbf{v}_k$ for $k = 1, \ldots, K$

Table 1 Time complexity of supervised clustering algorithms

Algorithm	Time complexity
MM-PL	$\mathcal{O}(NPL \log L)$
Kmeans-Mal	$\mathcal{O}(KN + NL)$
Kmeans-PL	$\mathcal{O}(KN + NL \log L)$
Naive-Kmeans	$\mathcal{O}(KN + KL \log L)$
Naive-RT	$\mathcal{O}(NL + NKL + NK \log N)$
ebms-RT	$\mathcal{O}(KNL^2 + NKL + NK \log N)$
K'om-RT	$\mathcal{O}(KNL^2 + NKL + NK \log N)$
RT-prune	$\mathcal{O}(NL \log N)$
RT-iter	$\mathcal{O}(NL + NKL + NK \log N)$

followed by finding the central ranking of examples in $S_1, S_2, \ldots, S_K$, which takes $\mathcal{O}(NL)$ using the modified Borda count (Cheng et al. 2009) and $\mathcal{O}(NL \log L)$ using the Plackett-Luce model. The complexity of the **Naïve**$_{Km}$ baseline is $\mathcal{O}(KN + KL \log L)$, as the L labels need to be sorted for K clusters following the K-means algorithm. The baseline algorithms **Naïve**$_{RT}$, **ebms**$_{RT}$ and K-**o'm**$_{RT}$ all train a Decision Tree in the second stage. Their time complexity only differs in the first stage of the algorithm where they search for the K central rankings. We note that the RankTree complexity with K possible ranking in leafs takes $\mathcal{O}(NKL + NK \log N)$ time, as we can calculate all possible distances $d_\tau(\pi_n, \rho_k)$ in $\mathcal{O}(NKL)$ time, followed by growing the tree using cumulative sums of different d_τ, which takes $\mathcal{O}(NK \log N)$ assuming tree depth of $\log N$. In general, **RT**$_{iter}$ has same time complexity as **Naïve**$_{RT}$ but it needs several iterations. The time complexity for **RT**$_{iter}$ is $\mathcal{O}(NL \log N)$ because for each possible split we need to find the central ranking ρ and the spread θ, which takes $\mathcal{O}(NL)$ time. When searching for the best split ρ for both sides can be updated instead of recalculated. Finally, the time complexity of the MM-PL algorithm is $\mathcal{O}(NPL \log L)$, where P is the total number of prototypes used, e.g., 100.

7 Experiments

In this section we evaluate the proposed algorithms for supervised clustering of label ranking data. Following the clustering experiments, we also evaluate the MM-PL algorithm in label ranking prediction task, where it was compared to several previously proposed algorithms. We describe the considered data sets in Sect. 7.1, while the experimental results are reported in Sect. 7.2.

In our preliminary experiments, we experimented with several different versions of the MM-PL algorithm. We were interested in comparing M-step ($v.1$) vs. M-step ($v.2$) as well as batch vs. stochastic prototype updates in M-step ($v.1$). The main conclusions are that the stochastic updates performed better than batch updates and that both $v.1$ and $v.2$ exhibited very similar performance. Following these results, in the following we report the performance of MM-PL $v.1$ with stochastic updates. Additional experiments with M-step ($v.2$) showed similar results to M-step ($v.1$), performing slightly better on some data sets and similar or slightly worse on others. The results of simulations that used the M-step ($v.2$) are not shown to avoid redundancy.

Let us first discuss MM-PL implementation details, specifically, initialization and parameter selection. The simplest way to initialize prototypes is to randomly place them in feature space or to select them by random sampling from the available training points. The label score vector $\mathbf{v}_k$ for each prototype was initialized by assigning $\epsilon \sim N(1, 0.1)$ to each label score. We have observed that setting all elements of label score vectors $\mathbf{v}_k$ to the same value leads to problems with the optimization procedure. The learning rate parameter γ was set to the initial value $\gamma_0 = 0.03$, and updated using $\gamma(n) = \gamma_0 \cdot \gamma_T/(\gamma_T + n)$, where $\gamma_T = 8N$ was used in all experiments. Parameter σ_p was initialized as the training data variance (Seo et al. 2003) and updated using the annealing schedule with $\sigma_T = 8N$. Training was terminated when $l(\lambda)$ (28) stopped decreasing by more than 10^{-4}.

7.1 Data description

We collected a total of 24 data sets for evaluation of MM-PL in both supervised clustering and label ranking prediction experiments. These data sets were used previously (Cheng et al. 2009, 2010) in evaluation of the label ranking prediction algorithms. Most of them were obtained by converting benchmark multi-class (A) and regression data sets (B) from the UCI and Statlog repositories into label ranking data, by using the following Naïve Bayes (A) and feature-to-label (B) (Cheng et al. 2009) techniques. For multi-class data sets, the Naïve Bayes classifier is first trained on the original data. Then, the label ranking data is obtained by sorting labels with respect to the predicted class probabilities. In case of a tie, the label with the lowest index is ranked first. For regression data sets, some attributes are removed from the data set, and each one is considered as a label. The removed attributes are standardized and then ordered by size to obtain label ranks. The data sets are available online.[3]

In addition, we used the real-world data from Hüllermeier et al. (2008) and sushi preference data from Kamishima and Akaho (2009). The data from the first source includes five data sets from the medical domain. The Yeast genome consists of 2,465 genes, each described by the phylogenetic profile of length 24. Same input features with different target rankings from five microarray experiments (spo, heat, dtt, cold, diau) resulted in five different data sets.

The sushi preference data collection[4] consists of data for several different preference learning tasks (collaborative filtering, object ranking and label ranking). The data is a result of a food-chain survey in which the users, described by 11 features, provided preferences for different sushi dishes in terms of a five-point-scale (for collaborative filtering) or full sushi rankings of 10 suggested sushis (for object and label ranking applications). In our

[3] http://www.uni-marburg.de/fb12/kebi/research/.

[4] http://www.kamishima.net/sushi/.

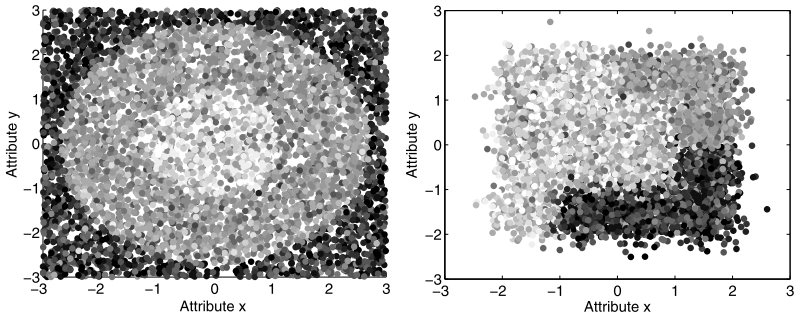
Table 2 Label ranking data sets: synthetic (S), semi-synthetic (SS) and real-world

Name	Domain	# instances N	# attributes d	# labels L
circles	synthetic	7,000	2	6
checker	synthetic	7,200	2	6
authorsh.	semi-synthetic (A)	841	70	4
iris	semi-synthetic (B)	150	4	3
bodyfat	semi-synthetic (B)	252	7	7
pendigits	semi-synthetic (A)	10,992	16	10
calhouse	semi-synthetic (B)	20,640	4	4
segment	semi-synthetic (A)	2,310	18	7
cpu-small	semi-synthetic (B)	8,192	6	5
stock	semi-synthetic (B)	950	5	5
elevators	semi-synthetic (B)	16,599	9	9
vehicle	semi-synthetic (A)	846	18	4
fried	semi-synthetic (B)	40,769	9	5
vowel	semi-synthetic (A)	528	10	11
glass	semi-synthetic (A)	214	9	6
wine	semi-synthetic (A)	178	13	3
housing	semi-synthetic (B)	506	6	6
wisconsin	semi-synthetic (B)	194	16	16
cold	biology	2,465	24	4
heat	biology	2,465	24	6
diau	biology	2,465	24	7
spo	biology	2,465	24	11
dtt	biology	2,465	24	4
sushi	food	5,000	11	10

experiments the data version from Kamishima and Akaho (2009) was used. The goal is to predict how a new customer ranks the 10 sushis based on his/her features. The additional features that describe each sushi were excluded since they could not be utilized in our model. The description of data sets is given in Table 2.

We also generated two 2-dimensional data sets to better characterize our model (Fig. 6). The real-life motivation for these synthetic data sets might be a country whose population consists of an unknown mixture of ethnically or culturally diverse people, each with different eating preferences. Based on data recorded at local stores, a food company is interested in spatially partitioning the country into K potentially irregularly shaped and nonadjacent regions, and creating a customized advertising strategy for each region. Such synthetic data was useful for visualization because the two features correspond to longitude and latitude. The observed behavior of different algorithms gives an insight into a general case where customers are described by multiple features. The two particular cases of interest were irregular-shaped (*circles* data) and unbalanced clusters (*checker* data).

In both data sets we assume there are 3 dominant central rankings with $L = 6$ labels (1: 123456, 2: 654321, 3: 316254). Let us refer to them as classes. Each point was assigned its class' label ranking corrupted by noise, such that with certain probability one or more labels switch places in total rank. The first data set (called *circles*) was created by uniformly



(a) Circles data set (b) Checkerboard data set

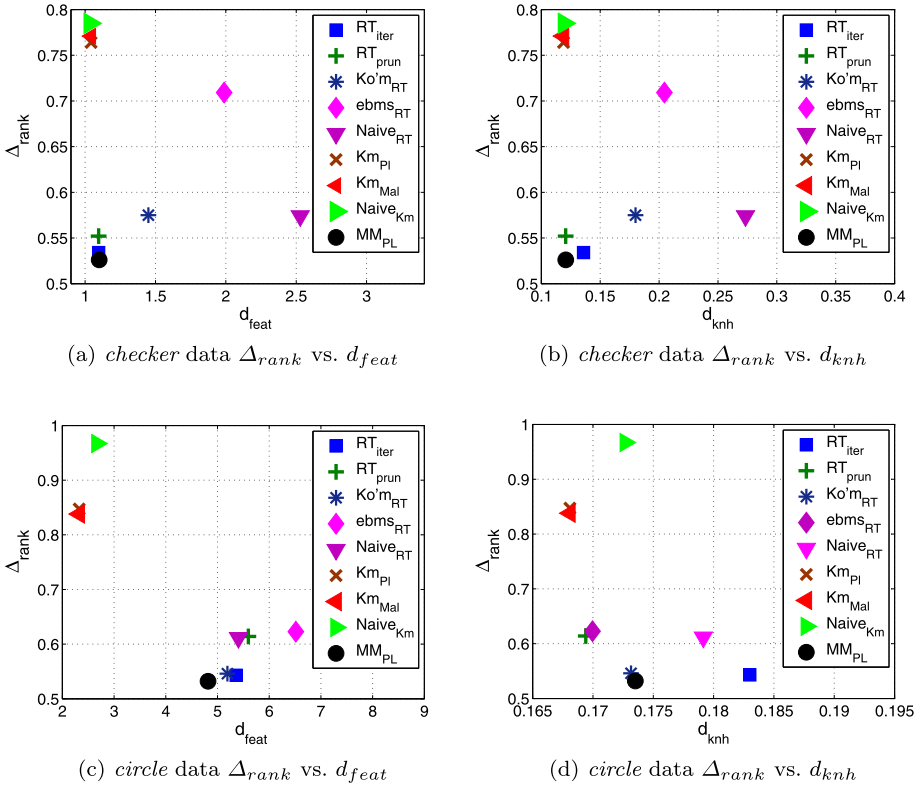
Fig. 6 Synthetic data sets (*color/gray* levels are based on Kendall distance from reference ranking $(1, 2, \ldots, L)$)

sampling 7,000 points from a square of width 6, centered at the origin. All points within distance 1.5 from origin were assigned to class 1, points within distances 1.5 to 2.7 were assigned to class 2, and the remaining points were assigned to class 3. Finally, for every point we add label ranking noise in the following way. One label is chosen at random, and the second label is chosen such that it is first label's first neighbor in the label ranking with probability 0.5, second neighbor with probability 0.3, third neighbor with probability 0.15 and fourth neighbor with probability 0.05 (if a label has only one neighbor we pick that neighbor as a second label, otherwise we pick one of two neighbors by throwing a fair coin). Then, we switch these two labels with probability 0.7, otherwise we quit noise-adding routine. If the noise was added we pick two new labels in the same way as before and switch them with probability 0.5, and if the second label switch occurred we choose two new labels and switch them with probability 0.3. The second data set (called *checker*) was generated in the following way. We sampled 450 points from each of 16 Gaussian distributions centered at the fields of 4×4 checkerboard, and assigned points from upper right and lower right Gaussians to classes 1 and 2, respectively, and the remaining points to class 3. Similarly to the first data set, we assigned each point its class' label ranking corrupted by noise. Matlab code to generate the data sets and other material, including implementation of all methods mentioned in this paper, are available upon request.

A common shortcoming of collection of user-content real-world label ranking data is missing preference information. Very often a user will only provide a partial rank of preferred items. Similarly to Cheng et al. (2009), we adopt the assumptions that labels are missing from rankings at random. Therefore, for each label in the ranking, we decide whether to discard it with a certain probability p_d. We did not pursue different directions when generating partial rankings, although other valid assumptions can be made, such as top-K labels or the assumption that only certain rankings get corrupted into partial rankings.

7.2 Experimental results

The clustering algorithms were compared using one external measure, Δ_{rank}, which evaluates cluster purity in label ranking space and two internal measures, d_{feat} and d_{knh}, which compactness in the feature space. To evaluate predictive performance, $loss_{LR}$ on test set was used. All results are averaged over 5 repetitions of a 10-fold cross-validation.



(a) *checker* data Δ_{rank} vs. d_{feat} (b) *checker* data Δ_{rank} vs. d_{knh}

(c) *circle* data Δ_{rank} vs. d_{feat} (d) *circle* data Δ_{rank} vs. d_{knh}

Fig. 7 Clustering performance comparison Δ_{rank} vs. d_{knh} and d_{feat}

7.2.1 Evaluation on synthetic data

The results on the synthetic *checker* and *circle* data sets are shown first. For this purpose, the number of clusters was set to $K = 3$ in order to test whether the correct clusters can be recovered. MM-PL used a total budget of $K \cdot P = 90$ prototypes, i.e., 30 per cluster. The resulting Δ_{rank} versus the resulting d_{feat} and d_{knh} for different algorithms are shown in Fig. 7. In addition, Tables 3 and 4 report which rankings are discovered as cluster centroids in different settings (complete rankings, 30 % and 60 % of missing labels) and the resulting partitions for some algorithms are visualized in Figs. 8 and 9. Finally, the performance in terms of $loss_{LR}$ summarized in the top two rows of Table 6.

We can conclude that MM-PL performed the best overall as it was able to uncover the actual clusters and the correct central rankings, even when the clusters were unbalanced (*checker*) and 60 % of labels were missing. Further, $\mathbf{RT}_{iter}$ algorithm performed very well and almost always found the correct central rankings, outperforming other RankTree-based algorithms, such as $\mathbf{RT}_{prun}$ and K-$\mathbf{o'm}_{RT}$ It can be observed that $\mathbf{Km}_{Mal}$ and $\mathbf{Km}_{Pl}$ underperformed when the label regions had complicated distributions, as they do not consider rankings in their first stage. $\mathbf{Naïve}_{RT}$ top K central ranking selection strategy did not prove efficient. Because of the imbalance in cluster sizes and the large amount of noise, it was not able to correctly uncover all the cluster central rankings by taking the top K occurring rankings as the central rankings. Neither did $\mathbf{ebms}_{RT}$. K-$\mathbf{o'm}_{RT}$ proved to be the best Rank

Table 3 Resulting cluster central rankings *checker* data set ($K = 3$)

	$\mathbf{Km}_{Mal}$	$\mathbf{Km}_{Pl}$	$\mathbf{naive}_{Km}$	$\mathbf{naive}_{RT}$	$\mathbf{ebms}_{RT}$	$\mathbf{Ko'm}_{RT}$	$\mathbf{RT}_{prun}$	$\mathbf{RT}_{iter}$	$\mathbf{MM}_{PL}$
full	**123456**	**123456**	**123456**	**123456**	132654	**123456**	132654	**123456**	**123456**
	123456	132456	**654321**	**654321**	653421	**654321**	**654321**	**654321**	**654321**
	361524	653421	423516	213456	653421	361245	**316254**	**316254**	**316254**
30 %	**123456**	**123456**	**123456**	**123456**	132546	132546	132654	**123456**	**123456**
	123456	345261	**654321**	**654321**	132645	**654321**	**654321**	**654321**	**654321**
	361254	312654	136254	124356	132645	214356	361254	361254	**316254**
60 %	**123456**	**123456**	**123456**	**123456**	132645	214536	123546	**123456**	**123456**
	316254	316254	**654321**	124356	132645	546321	653421	**654321**	**654321**
	316452	123456	123645	123546	132546	132645	**316254**	361254	**316254**

Table 4 Resulting cluster central rankings *circle* data set ($K = 3$)

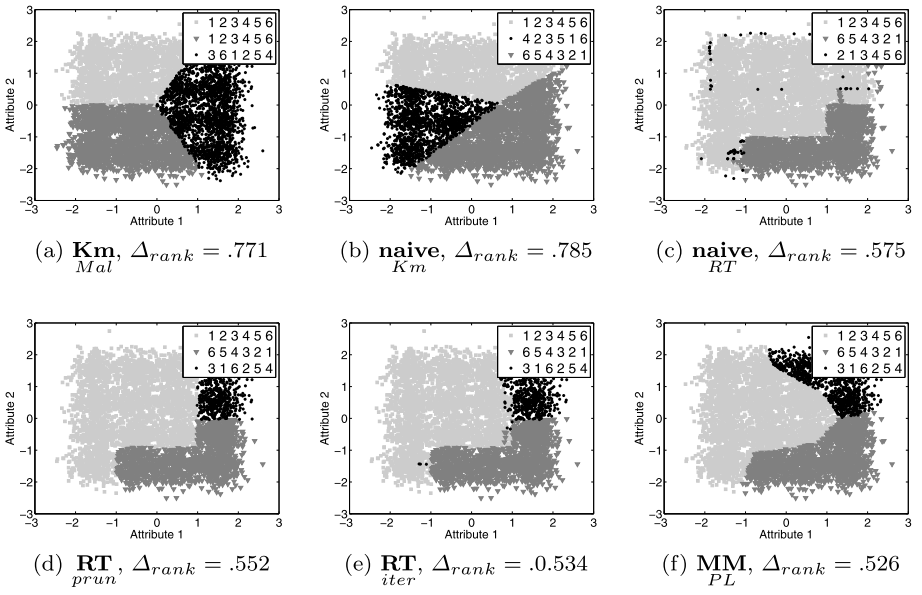
	$\mathbf{Km}_{Mal}$	$\mathbf{Km}_{Pl}$	$\mathbf{naive}_{Km}$	$\mathbf{naive}_{RT}$	$\mathbf{ebms}_{RT}$	$\mathbf{Ko'm}_{RT}$	$\mathbf{RT}_{prun}$	$\mathbf{RT}_{iter}$	$\mathbf{MM}_{PL}$
full	361254	361254	**123456**	136254	**123456**	**123456**	132465	**123456**	**123456**
	316254	361254	**654321**	**654321**	132546	**654321**	**654321**	**654321**	**654321**
	361524	365124	**316254**	**316254**	312654	**316254**	361254	**316254**	**316254**
30 %	361524	365124	132654	312654	653421	124356	132456	**123456**	**123456**
	361254	361254	**654321**	**654321**	653421	**654321**	**654321**	**654321**	**654321**
	361524	361254	654321	**316254**	132654	**316254**	361254	**316254**	**316254**
60 %	361524	361524	635124	316524	316254	124356	132546	**123456**	**123456**
	361254	361254	316254	361254	316254	635412	653421	**654321**	**654321**
	361524	361254	**316254**	**316254**	**316254**	126354	361254	**316254**	**316254**

Clustering → FR. It found the correct central rankings for *circle* data set in case of complete label rankings (Table 3). However, it was less effective when label preference information was missing. In case of *checker* data K**-o'm**$_{RT}$ found clusters that are compact in label space (low Δ_{rank}), however it resulted in poor predictive performance. It assigned to the smallest cluster some examples from other two clusters based on distances in label ranking space. Finally, **Naïve**$_{Km}$ was not as successful as other algorithms on both data sets.

In Fig. 7 we show the trade-off between compactness in feature space (measured by d_{feat} and d_{knh}) and purity in label ranking space (measured by Δ_{rank}). The best algorithm is the one in the bottom left corner. We can conclude that the best algorithms overall were MM-PL and $\mathbf{RT}_{iter}$ as they achieved the best trade-off. It can also be observed that $\mathbf{Km}_{Mal}$ and $\mathbf{Km}_{Pl}$ found the most compact clusters in feature space, especially when measured by d_{feat} as they directly optimize for this measure. The gap between $\mathbf{Km}_{Mal}$, $\mathbf{Km}_{Pl}$ and other algorithms decreases when we look at the d_{knh} measure in the case of *circle* data, as d_{feat} is known to have high values for non-globular clusters, such as circles in our case.

Overall, as the percentage of missing labels increased, all algorithms, except MM-PL and to some extend RT_{iter}, had more trouble finding the correct cluster central rankings.

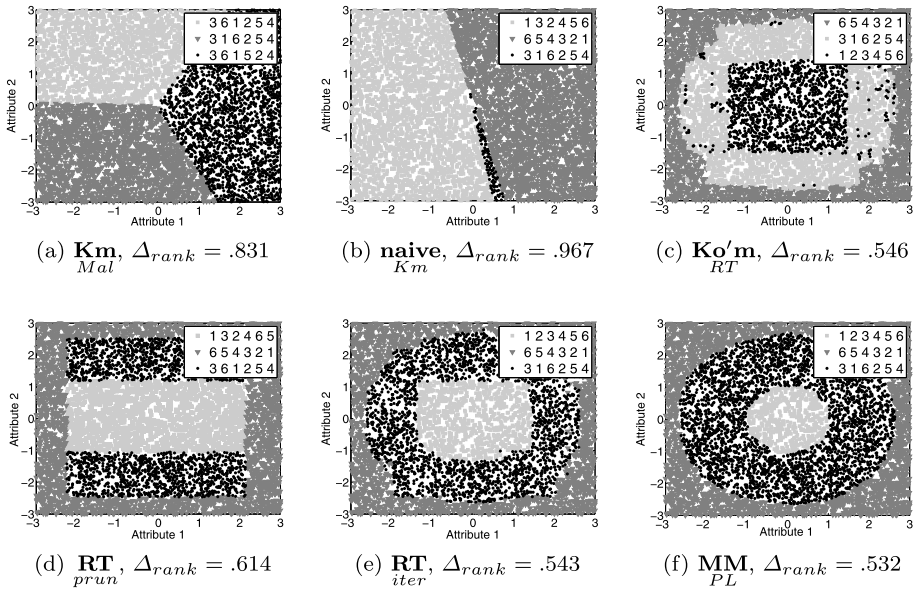
It should be noted that K-means based algorithms are likely to find different central rankings depending of the initial seeding. On the other hand, the other algorithms provide more stable clusterings.



(a) $\underset{Mal}{\mathbf{Km}}$, $\Delta_{rank} = .771$ (b) $\underset{Km}{\mathbf{naive}}$, $\Delta_{rank} = .785$ (c) $\underset{RT}{\mathbf{naive}}$, $\Delta_{rank} = .575$

(d) $\underset{prun}{\mathbf{RT}}$, $\Delta_{rank} = .552$ (e) $\underset{iter}{\mathbf{RT}}$, $\Delta_{rank} = .0.534$ (f) $\underset{PL}{\mathbf{MM}}$, $\Delta_{rank} = .526$

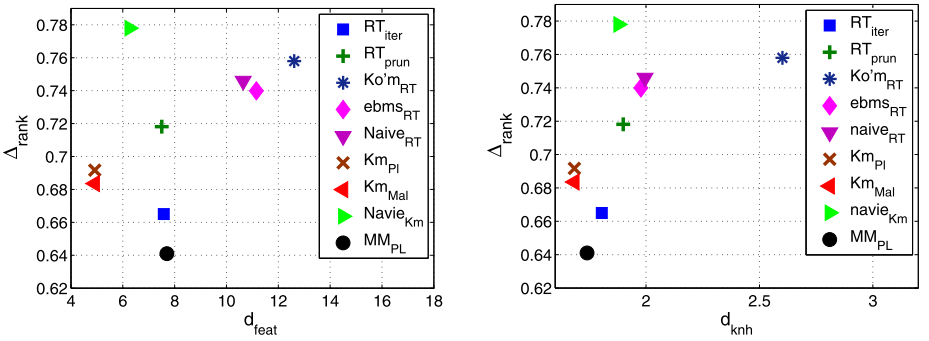
Fig. 8 Clustering performance comparison on *checker* data set ($K = 3$)



(a) $\underset{Mal}{\mathbf{Km}}$, $\Delta_{rank} = .831$ (b) $\underset{Km}{\mathbf{naive}}$, $\Delta_{rank} = .967$ (c) $\underset{RT}{\mathbf{Ko'm}}$, $\Delta_{rank} = .546$

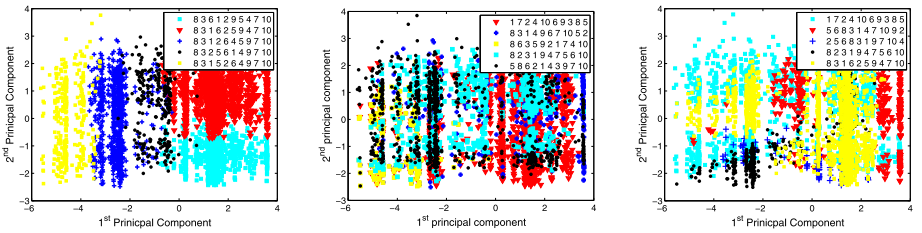
(d) $\underset{prun}{\mathbf{RT}}$, $\Delta_{rank} = .614$ (e) $\underset{iter}{\mathbf{RT}}$, $\Delta_{rank} = .543$ (f) $\underset{PL}{\mathbf{MM}}$, $\Delta_{rank} = .532$

Fig. 9 Clustering performance comparison on *circle* data set ($K = 3$)



(a) *sushi* data Δ_{rank} vs. d_{feat} (b) *sushi* data Δ_{rank} vs. d_{knh}

Fig. 10 *sushi* data performance comparison Δ_{rank} vs. d_{knh} and d_{feat}



(a) $\underset{Mal}{\mathbf{Km}}$, $\Delta_{rank} = .695$ (b) $\underset{RT}{\mathbf{Ko'm}}$, $\Delta_{rank} = .758$ (c) $\underset{PL}{\mathbf{MM}}$, $\Delta_{rank} = .641$

Fig. 11 Clustering performance comparison on *sushi* data set ($K = 5$)

7.2.2 Evaluation on semi-synthetic and real-world data

Next, we evaluated the algorithms on the real-world sushi data set with $K = 5$ clusters. In Fig. 10 we shown the performance in terms of achieved Δ_{rank} vs. d_{feat} and Δ_{rank} vs. d_{knh} trade-offs. The predictive performance in terms of $loss_{LR}$ is shown at the bottom row of Table 6. Finally, Table 5 shows the central rankings found by each algorithm and Fig. 11 visualizes the found clusters for some algorithms by projecting features to the first two principal components.

We can make several observations:

(1) $\mathbf{Km}_{Mal}$, $\mathbf{Km}_{Pl}$ and $\mathbf{Naïve}_{Km}$ clusters were compact in feature space (d_{feat} and d_{knh}) but less compact in label space (Δ_{rank})
(2) Rank Clustering → FR were compact in the label space (Δ_{rank}), but far less compact in the feature space (d_{feat} and d_{knh})
(3) MM-PL clusters were compact in both label and feature space
(4) $\mathbf{RT}_{iter}$ outperformed other RankTree-based methods and came fairly close to MM-PL

For example, even though $\mathbf{Km}_{Mal}$ partitioned the feature space well, the partitions were not descriptive enough as all cluster central rankings had sushis 8 and 1 at the first two positions and 7 and 10 at the last two positions. Thus, the cluster-specific promotional material would only differ in the mid part of customer preferences. Due to the fact that cluster members were scattered around the feature space (Fig. 11b), K**-o'm**$_{RT}$ resulted in poor gen-

Table 5 Sushi resulting cluster central rankings ($K = 5$), no missing labels

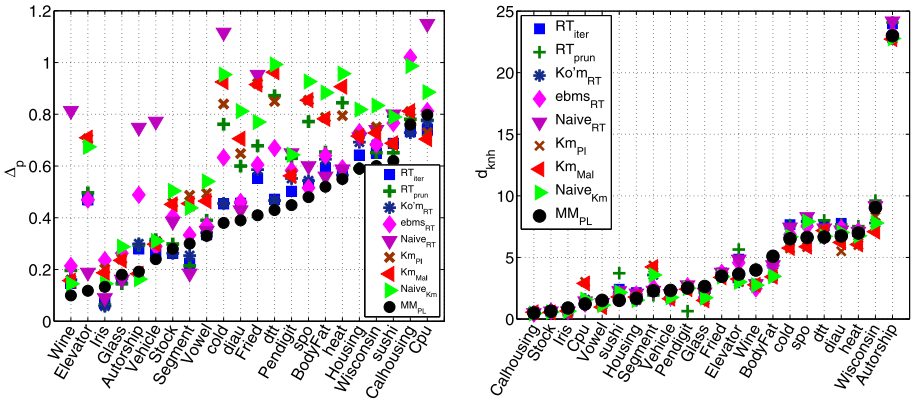
1 Rank	$\mathbf{Km}_{Mal}$	$\mathbf{Km}_{Pl}$	$\mathbf{naive}_{Km}$	$\mathbf{naive}_{RT}$	$\mathbf{ebms}_{RT}$	$\mathbf{Ko'm}_{RT}$	$\mathbf{RT}_{prun}$	$\mathbf{RT}_{iter}$	$\mathbf{MM}_{PL}$
8316259471<u>0</u>	8361295471<u>0</u>	8362159471<u>0</u>	5218643971<u>0</u>	2568319710<u>4</u>	8314972106<u>5</u>	8231947561<u>0</u>	8156392471<u>0</u>	2568139710<u>4</u>	1724106938<u>5</u>
	8316259471<u>0</u>	8316924571<u>0</u>	1742310986<u>5</u>	5683149271<u>0</u>	2813596471<u>0</u>	8314967105<u>2</u>	8324167910<u>5</u>	8361295471<u>0</u>	5683147109<u>2</u>
	8361295471<u>0</u>	8316294571<u>0</u>	8391264710<u>5</u>	8365921410<u>7</u>	8635921471<u>0</u>	8635921741<u>0</u>	8563214971<u>0</u>	5683147910<u>2</u>	2568319710<u>4</u>
	8325614971<u>0</u>	8321654971<u>0</u>	8526391741<u>0</u>	8562139410<u>7</u>	5861423971<u>0</u>	5862143971<u>0</u>	8615349721<u>0</u>	8324167910<u>5</u>	8231947561<u>0</u>
	8315264971<u>0</u>	8312496571<u>0</u>	8536149271<u>0</u>	1839247106<u>5</u>	8635921471<u>0</u>	1724106938<u>5</u>	8625391471<u>0</u>	8316259471<u>0</u>	8316259471<u>0</u>
$\Delta_r = .726$	.695	.692	.778	.745	.739	.758	.718	.665	**.641**

Table 6 Label Ranking Supervised Clustering performance comparison in terms of $loss_{LR}$ (complete rankings)

Data	1 Rank	$\mathbf{Km}_{Mal}$	$\mathbf{Km}_{Pl}$	$\mathbf{naive}_{Km}$	$\mathbf{naive}_{RT}$	$\mathbf{ebms}_{RT}$	$\mathbf{Ko'm}_{RT}$	$\mathbf{RT}_{prun}$	$\mathbf{RT}_{iter}$	$\mathbf{MM}_{PL}$
checker	.398	.340	.343	.296	.291	.348	.269	.268	.263	**.250**
circle	.393	.392	.392	.444	.263	.281	.260	.292	.258	**.253**
fried	.493	.413	.414	.334	.273	.280	.276	.299	.238	**.226**
calhouse	.380	.364	.364	.424	.335	.333	.333	.346	**.331**	.349
elevators	.435	.347	.333	.300	.213	.211	.211	.217	.211	**.171**
pendigits	.383	.290	.292	.308	.303	.294	.283	.264	.255	**.172**
cpu-small	.431	.362	.369	.374	.353	.403	.354	.346	**.337**	.356
segment	.419	.214	.221	.188	.153	.168	.145	.129	**.112**	.146
wisconsin	.485	.451	.456	.443	.462	.449	.451	.459	.441	**.433**
vowel	.334	.253	.268	.274	.211	.207	.208	.202	**.194**	.209
vehicle	.335	.132	.125	.113	.101	.123	.102	.106	.092	**.090**
stock	.445	.213	.213	.218	.169	.200	.168	.170	**.138**	.140
iris	.451	.091	.096	.059	**.056**	.096	**.056**	.058	**.056**	.061
glass	.171	.110	.113	.113	.110	.114	**.100**	.106	.104	.108
authorsh	.269	.072	.075	**.066**	.143	.178	.083	.115	.113	.079
bodyfat	.450	.451	.453	.454	.448	.451	.450	.448	.448	**.430**
wine	.234	.050	**.044**	.058	.082	.110	.080	.079	.079	.047
housing	.412	.376	.384	.407	.400	.388	.364	.372	.358	**.339**
cold	.401	.395	.395	.408	.413	.391	.410	.413	.404	**.386**
heat	.464	.438	.440	.453	.457	.455	.457	.463	.453	**.433**
diau	.348	.336	.340	.385	.380	.356	.367	.367	.345	**.327**
dtt	.438	.416	.420	.432	.435	.425	.437	.439	.431	**.410**
spo	.441	**.431**	.438	.440	.467	.448	.452	.461	.448	**.431**
sushi	.392	.372	.342	.383	.375	.373	.380	.339	.336	**.332**
average	.392	.304	.305	.307	.287	.295	.279	.282	.268	**.257**
avg. rank	9.04	5.37	6.00	6.96	5.83	6.00	4.58	5.21	2.67	**2.21**
time (sec)	22.8	10.1	12.5	9.3	1K	8.3K	2.1K	532	2.4K	234

eralization (Table 6). MM-PL clusters were informative and diverse in label space and also consistently distributed in the feature space, allowing for a simpler and more intuitive rule when assigning new customers to clusters. For this reason it had the best overall predictive performance (Table 6).

The results on the remaining data sets are reported in terms of $Loss_{LR}$ in Table 6, and in terms of Δ_{rank} and d_{knh} in Figs. 12a and 12b. We also show the aggregated average Δ_{rank} vs. d_{knh} trade-off across all data sets in Fig. 13. The number of clusters was set to $K = 10$. Therefore, $\mathbf{RT}_{prun}$ was pruned until the leafs contain at most 10 different rankings, and the total budget size of MM-PL was set to $P \cdot K = 100$, i.e., 10 prototypes per cluster. Bottom rows of Table 6 report average $Loss_{LR}$ on all data sets as well as the average rank of algorithms. For each data set, the algorithms were ranked in decreasing order of performance and the average rankings across all data sets are reported. Finally, the running times (number



(a) Purity Δ_{rank} (b) Compactness d_{knh}

Fig. 12 Label Ranking Supervised Clustering performance comparison (complete rankings)

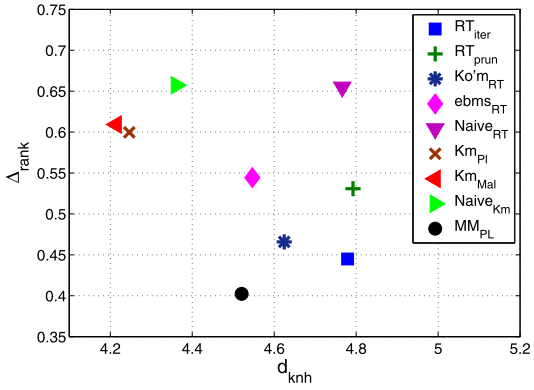


Fig. 13 Aggregate Δ_{rank} vs. d_{knh} performance summary (complete rankings)

of seconds it takes to complete a single 5-fold cross-validation) averaged over all data sets are reported at the bottom of Table 6.

The results include 1-Rank algorithm result that serves as an indicator of how well we can represent data using a single label rank.

We can make several conclusions regarding the performance of MM-PL. Firstly, by observing Fig. 12a, we can conclude that in the most cases (17 out of 24) MM-PL resulted in the purest clusters in the label ranking space. Secondly, MM-PL averaged the 4-th best compactness in the feature space (Fig. 12b), performing close to $\mathbf{Km}_{Mal}$, $\mathbf{Km}_{Pl}$ and $\mathbf{Naïve}_{Km}$. Thirdly, MM-PL achieved the best Δ_{rank} vs. d_{knh} trade-off overall (Fig. 13). Finally by observing the results in Table 6 we can conclude that MM-PL had the best predictive performance as well (ranked best 15 out of 24 times) with minimum average rank of 2.21.

Regarding the running time, MM-PL and was ranked 4-th best out of all clustering algorithms (excluding 1 Rank), averaging 234 seconds per 5-fold cross-validation. Expectedly, the largest difference in running times was observed on data sets with large L and large N. For example, running times for Wisconsin data set which has small number of observations $N = 194$ but large number of labels $L = 16$, were as follows: 270.21 sec ($\mathbf{RT}_{iter}$), 95 sec ($\mathbf{RT}_{prun}$), 23.4 sec ($\mathbf{Km}_{Mal}$), 423.3 sec ($\mathbf{ebms}_{RT}$), 48.2 sec MM-PL. Also, for Fried data set with $N = 40{,}769$ and $L = 5$, the running times were: 3.5K sec ($\mathbf{RT}_{iter}$), 1.6K sec ($\mathbf{RT}_{prun}$), 252 sec ($\mathbf{Km}_{Mal}$), 3K sec ($\mathbf{Ko'm}_{RT}$), 632 sec MM-PL.

$\mathbf{RT}_{iter}$ was ranked the second best algorithm overall, outperforming other RT solutions, including $\mathbf{RT}_{prun}$ and $\mathbf{Ko'm}_{RT}$. However, it comes at a cost of large computational time. It was slower than other RT-based methods and much slower than MM-PL due to its iterative nature and the fact that it needs to calculate the sum of losses for every possible split. The speed-up at probable cost of performance could be achieved by using greedy solutions that do not consider each split. This holds for all RT-based methods.

Depending on the data properties, various algorithms showed their advantages. If the clusters in feature space corresponded to clusters in the label space $\mathbf{Km}_{Mal}$, $\mathbf{Km}_{Pl}$ performed well, especially $\mathbf{Km}_{Pl}$ which was ranked high in experiments with missing preferences (Table 8). If top K label rankings matched the true cluster centroids $\mathbf{Naive}_{RT}$ was a good fit.

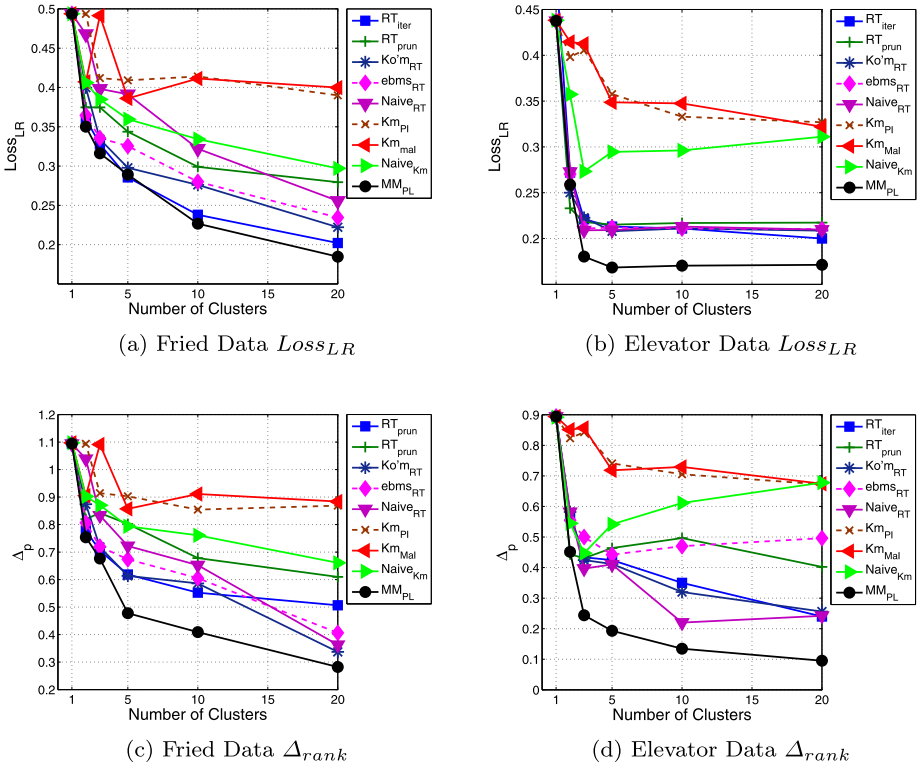
Regarding the performance of the Rank Clustering → FR algorithms, it depended greatly on the choice of the K rankings, which are found in the first stage using **Naive**, **ebms** or the **Ko'm** strategy. **Ko'm** proved to be the best choice and $\mathbf{Ko'm}_{RT}$ was ranked as the third best algorithm overall. Surprisingly, **Naive** did slightly better than **ebms**, except on real-world data sets, data sets with large L and the cases with missing labels (shown in Table 8). In those cases **ebms** outperformed **Naive**. The reason for this behavior may be in the fact that **ebms** strategy is specifically designed for large number of labels (Meilă and Bao 2010). Regarding the computational complexity, in some cases Rank Clustering → FR had lower running times when compared to $\mathbf{RT}_{prun}$ and $\mathbf{RT}_{iter}$, due to the fact that they know the candidate K rankings in advance, and do not need to find central ranking for each possible split. However, $\mathbf{RT}_{prun}$ and $\mathbf{RT}_{iter}$ do not have a pre-processing stage of finding K rankings, which is costly in some cases, especially for **ebms**, which was very slow on large data sets and was overall the slowest algorithm.

In Fig. 14 we evaluate the performance of label rank clustering algorithms with respect to the number of clusters K. The investigation was performed on *Fried* and *Elevator* data sets. We were interested in both the performance when small number of clusters is specified, which is important when working on a limited budget, and at what point increasing the number of clusters does not further improve performance. Performance was evaluated based on both $Loss_{LR}$ (top figures) and Δ_{rank} (bottom figures).

We can observe that the change in performance from $K = 1$ to $K = 2$ was already significant. Performance on *Fried* data set continued to improve after $K = 20$ clusters, while on *Elevator* data set it stopped improving after $K = 10$, i.e., clusters became no more pure when increasing K.

In the next set of experiments we evaluated the algorithms in the missing label scenario. These results are reported in Tables 7 and 8. Missing label scenario was simulated using the following procedure. First we decided on the percentage of missing labels p_d, e.g., $p_d = 30\ \%$. Then, for each example in the training data we flipped a biased coin for every label in the ranking to decide whether to delete that label (with probability p_d).

In Tables 7 and 8 we show results for 30 % and 60 % missing labels. It is interesting to observe that the dominance of MM-PL increased when label information is missing. Its average rank significantly improved, as it is now the best performing algorithm on almost all data sets, which was not the case before the removal of labels. Its rank improved from 2.21 with no missing labels to 1.75 with 30 % and 1.33 with 60 % missing labels. It can also be observed that the rank of $\mathbf{Km}_{Pl}$ algorithm improves with the percentage of missing labels. For example, in experiments with no missing labels, it shares the 7-th position with $\mathbf{ebms}_{RT}$, but is ranked 3-rd overall in experiments with 60 % missing labels. Further, the rank of $\mathbf{ebms}_{RT}$ also improves with the percentage of missing labels, while the rank of $\mathbf{Ko'm}_{RT}$ worsens. As a result, in experiments with 60 % missing labels, $\mathbf{ebms}_{RT}$ is ranked



(a) Fried Data $Loss_{LR}$ (b) Elevator Data $Loss_{LR}$

(c) Fried Data Δ_{rank} (d) Elevator Data Δ_{rank}

Fig. 14 Clustering performance comparison with different K

better than $\mathbf{Naive}_{RT}$ and $\mathbf{Ko'm}_{RT}$, which was not the case in experiments with full rankings. $\mathbf{RT}_{iter}$ remains the second best algorithm, while $\mathbf{RT}_{prun}$ holds the third overall position.

In Fig. 15 we evaluate the performance of label rank clustering algorithms with respect to the percentage of missing label information (p ranging from 10 % to 90 %) on *Fried* and *Elevator* data sets.

This investigation is of great practical importance. For example, a company might have a large number of products, but its survey data could contains ranks for only a small subset of products for each customer. Therefore, the percentage of missing labels might easily be as high as 90 %.

It can be observed that MM-PL and $\mathbf{RT}_{iter}$ were much more robust to missing information than baseline algorithms. $\mathbf{Km}_{Mal}$ and $\mathbf{Naive}_{Km}$ performed equally poor for $p = 10$ % to 70 % and started to degrade even more afterwards. When compared to them $\mathbf{Km}_{Pl}$ performance did not change much even when 90 % of labels are missing. Rank Clustering → FR algorithms' performance suddenly dropped, as soon as $p = 60$ %, and continued dropping rapidly, where the least degradation was observed with $\mathbf{ebms}_{RT}$. Further $\mathbf{RT}_{prun}$ had better performance than Rank Clustering → FR algorithms. Moreover, performance of MM-PL and $\mathbf{RT}_{iter}$ decreased much more gradually when compared to other algorithms. Finally, MM-PL performance was satisfactory even when $p = 70$ %, which makes it a good choice for supervised clustering of label ranking data with missing label information.

Table 7 Label Ranking Supervised Clustering performance comparison in terms of $loss_{LR}$ (incomplete rankings—30 % missing labels)

Data	$\mathbf{Km}_{Mal}$	$\mathbf{Km}_{Pl}$	$\mathbf{naiv}_{Km}$	$\mathbf{naiv}_{RT}$	$\mathbf{ebms}_{RT}$	$\mathbf{Kom}_{RT}$	$\mathbf{RT}_{prn}$	$\mathbf{RT}_{itr}$	$\mathbf{MM}_{PL}$
checker	.340	.336	.300	.282	.316	.257	.292	.254	**.249**
circle	.392	.381	.410	.281	.303	.263	.294	.260	**.258**
fried	.412	.408	.320	.305	.278	.279	.283	.260	**.236**
calhousing	.365	.364	.463	.346	.350	.344	.350	.339	.352
elevator	.334	.330	.301	.218	.218	.216	.220	.218	**.210**
pendigits	.289	.289	.287	.305	.302	.286	.266	.243	**.181**
cpu-small	.363	.367	.389	.362	.366	.361	.358	**.350**	.356
segment	.214	.223	.197	.180	.175	.167	.145	**.136**	.145
wisconsin	.455	.456	.443	.481	.445	.458	.448	.444	**.435**
vowel	.255	.260	.276	.238	.220	.226	.210	**.208**	.211
vehicle	.131	.125	.165	.110	.140	.098	.099	.100	**.095**
stock	.217	.217	.227	.176	.220	.195	.179	.183	**.148**
iris	.093	.088	.142	**.079**	.131	.087	.090	.088	.088
glass	.117	.114	.173	.125	.115	.116	**.110**	**.110**	.114
authorship	.073	**.072**	**.072**	.128	.180	.134	.117	.110	.075
bodyfat	.455	.453	.459	.459	.447	.443	.454	.452	**.436**
wine	.051	.049	.163	.088	.128	.077	.076	.062	**.045**
housing	.377	.375	.421	.389	.390	.391	.370	.358	**.357**
cold	.405	.397	.404	.414	.406	.427	.416	.400	**.393**
heat	.440	.442	.451	.457	.456	.459	.462	.455	**.437**
diau	.339	**.338**	.376	.350	.357	.363	.361	.344	**.338**
dtt	.422	.420	.428	.441	.432	.442	.452	.438	**.417**
spo	.434	.439	.447	.454	.450	.455	.456	**.429**	.435
sushi	.369	.342	.374	.367	.353	.378	.353	.342	**.335**
average	.306	.304	.320	.293	.299	.288	.286	.274	**.264**
rank	5.96	5.00	6.96	5.75	5.83	5.33	4.96	2.75	**1.75**
time (sec)	13.5	15.8	10.9	1.5K	10.1K	2.4K	486	2.5K	228

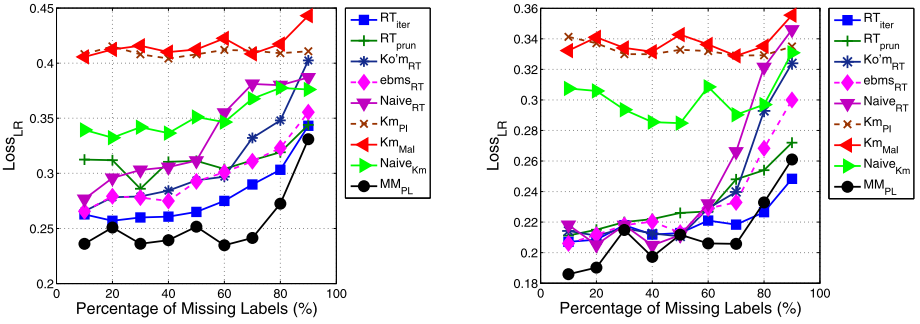
8 Conclusion

Although it has many potential real-world applications, supervised clustering of complex label rank data has not been heavily studied. We established several heuristic baselines for this problem and proposed two principled algorithms, the Plackett-Luce (PL) Mixture Model and the Iterative RankTree model specifically tailored for this application. We experimentally showed the strength of the PL and RT_{iter} models by conducting extensive experiments on two synthetic, sixteen semi-synthetic, and six real-world data sets.

Depending on the data properties, various algorithms showed their advantages. For example, when the clusters in the feature space corresponded to clusters in the label space the $\mathbf{Km}_{Mal}$ and $\mathbf{Km}_{Pl}$ baseline approaches performed well. If clusters were evenly distributed in the label space and most occurring ranks matched cluster central rankings the $\mathbf{Ko'm}_{RT}$ baseline approach performed well. However, in case of unbalanced clusters, either in label

Table 8 Label Ranking Supervised Clustering performance comparison in terms of $loss_{LR}$ (incomplete rankings—60 % missing labels)

Data	$\mathbf{Km}_{Mal}$	$\mathbf{Km}_{Pl}$	$\mathbf{naiv}_{Km}$	$\mathbf{naiv}_{RT}$	$\mathbf{ebms}_{RT}$	$\mathbf{Kom}_{RT}$	$\mathbf{RT}_{prn}$	$\mathbf{RT}_{itr}$	$\mathbf{MM}_{PL}$
checker	.342	.340	.308	.413	.403	.325	.300	.268	**.260**
circle	.392	.382	.396	.362	.379	.337	.295	.263	**.261**
fried	.422	.412	.395	.355	.301	.297	.299	.275	**.251**
calhousing	.365	.364	.501	.367	.362	.358	.356	.354	**.349**
elevator	.336	.332	.308	.238	.229	.229	.227	.221	**.214**
pendigits	.291	.291	.337	.333	.317	.324	.270	.258	**.202**
cpu-small	.365	.366	.413	.469	.379	.383	.364	**.359**	.360
segment	.213	.217	.229	.208	.206	.238	.160	.157	**.156**
wisconsin	.465	.472	.461	.490	.463	.467	.464	.453	**.444**
vowel	.265	.265	.314	.277	.244	.271	.245	**.233**	.242
vehicle	.136	.134	.210	.127	.146	.135	.118	.114	**.109**
stock	.222	.222	.259	.233	.252	.256	.195	.200	**.162**
iris	.136	.134	.188	.210	.208	.215	.194	.138	**.127**
glass	.139	.134	.211	.127	.138	.156	.139	**.126**	.130
authorship	.082	**.075**	.077	.143	.214	.163	.138	.129	.082
bodyfat	.469	.465	.464	.466	.467	.480	.467	.453	**.444**
wine	.087	.078	.220	.145	.149	.154	.151	.137	**.076**
housing	.397	.396	.436	.399	.408	.415	.384	.370	**.365**
cold	.404	.404	.418	.419	.406	.445	.445	.438	**.398**
heat	.447	.445	.456	.458	.472	.467	.472	.454	**.439**
diau	.345	**.340**	.378	.357	.357	.383	.382	.345	.343
dtt	.424	**.422**	.432	.445	.434	.457	.458	.448	**.422**
spo	.443	**.440**	.471	.467	.458	.462	.465	.461	**.440**
sushi	.379	**.341**	.394	.368	.354	.388	.390	.350	.343
average	.315	.311	.345	.328	.323	.325	.319	.292	**.276**
rank	5.17	4.08	6.96	6.33	5.67	6.67	5.42	2.96	**1.33**
time (sec)	13.8	16.4	11.6	1.4K	10K	2.2K	460	2.6K	188



(a) Fried Data ($L = 5$) (b) Elevator Data ($L = 9$)

Fig. 15 Clustering performance comparison with different percent of missing labels

or feature space, or presence of partial rankings, the baseline approaches did not perform as well as the MM-PL and RT_{iter} models.

The MM-PL model achieved state-of-the-art clustering performance, showing better adaptivity to missing information and smaller number of clusters than the RT_{iter} model. When faced with the missing label problem, and particularly when a large fraction of labels are missing, the MM-PL model works exceptionally well. In addition to the impressive accuracy, the PL model has a small memory footprint, making it an attractive option in memory-constrained scenarios. It also has favorable time complexity, scaling linearly in the size of data set and number of prototypes, which results in very fast and efficient model. These features make MM-PL an attractive option for label ranking.

The new application of supervised clustering opens many interesting problems. For instance, an interesting direction would be to study supervised clustering of label ranking data from a Bayesian perspective, since maximum likelihood may lead to overfitting for sparse label ranking data. It would also be interesting to investigate and compare different assumptions of missing preferences, such as top-K assumptions, constant corruption, and missing at random. Our future work also includes addressing current shortcomings of the MM-PL model, including its inability to model arbitrary pairwise preferences which do not form a ranking. This could be solved by developing a mixture model with a pair-wise probabilistic model instead of the PL model.

Acknowledgements We are grateful to Marina Meilă for providing software for the EBMS algorithm, and to the anonymous reviewers for constructive feedback and insightful suggestions which greatly improved this manuscript.

References

Bradley, R. A., & Terry, M. E. (1952). Rank analysis of incomplete block designs. I. The method of paired comparisons. *Biometrika*, *39*, 324–345.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees*. Belmont: Wadsworth.

Brun, M., Sima, C., Hua, J., Lowey, J., Carroll, B., Suh, E., & Dougherty, E. R. (2007). Model-based evaluation of clustering validation measures. *Pattern Recognition*, *40*(3), 807–824.

Cheng, W., Hühn, J., & Hüllermeier, E. (2009). Decision tree and instance-based learning for label ranking. In *International conference on machine learning* (pp. 161–168).

Cheng, W., Dembczyński, K., & Hüllermeier, E. (2010). Label ranking methods based on the Plackett-Luce model. In *International conference on machine learning* (pp. 215–222).

Coppersmith, D., Fleischer, L. K., & Rudra, A. (2006). Ordering by weighted number of wins gives a good ranking of weighted tournaments. In *ACM-SIAM symposium on discrete algorithms* (pp. 776–782).

de Borda, J. C. (1781). Memoire sur les Élections au Scrutin. *Hist. Acad. R. Sci.*

Dekel, O., Manning, C., & Singer, Y. (2003). *Advances in neural information processing systems: Vol. 16. Log-linear models for label ranking*. Cambridge: MIT Press.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B. Methodological*, *39*, 1–38.

Domshlak, C., Hüllermeier, E., Kaci, S., & Prade, H. (2011). Preferences in AI: an overview. *Artificial Intelligence*, *175*(7–8), 1037–1052.

Dwork, C., Kumar, R., Naor, M., & Sivakumar, D. (2001). Rank aggregation methods for the web. In *International conference on world wide web* (pp. 613–622). New York: ACM.

Eick, C. F., Vaezian, B., Jiang, D., & Wang, J. (2006). Discovery of interesting regions in spatial data sets using supervised clustering. In *Knowledge discovery in databases* (pp. 127–138). Berlin: Springer.

Eick, C., Zeidat, N., & Zhao, Z. (2011). Supervised clustering—algorithms and benefits. In *International conference on tools with AI* (pp. 774–776).

Finley, T., & Joachims, T. (2005). Supervised clustering with support vector machines. In *International conference on machine learning* (pp. 217–224). New York: ACM.

Gärtner, T., & Vembu, S. (2010). Label ranking algorithms: a survey. In J. Fürnkranz & E. Hüllermeier (Eds.), *Preference learning* (pp. 45–64). Berlin: Springer.

Grbovic, M., & Vucetic, S. (2009). Regression learning vector quantization. In *International conference on data mining* (pp. 788–793).
Guiver, J., & Snelson, E. (2009). Bayesian inference for Plackett-Luce ranking models. In *International conference on machine learning* (pp. 377–384). New York: ACM.
Han, J., & Kamber, M. (2001). *Data mining: concepts and techniques*. San Mateo: Morgan Kaufmann.
Har-Peled, S., Roth, D., & Zimak, D. (2003). Constraint classification for multiclass classification and ranking. In *Advances in neural information processing systems* (pp. 785–792). Cambridge: MIT Press.
Hüllermeier, E., Fürnkranz, J., Cheng, W., & Brinker, K. (2008). Label ranking by learning pairwise preferences. *Artificial Intelligence*, *172*, 1897–1916.
Hunter, D. R. (2004). MM algorithms for generalized Bradley-Terry models. *The Annals of Statistics*, *32*, 384–406.
Jain, A. K., & Dubes, R. C. (1988). *Algorithms for clustering data*. New York: Prentice Hall.
Kamishima, T., & Akaho, S. (2009). Efficient clustering for orders. In *Mining complex data* (pp. 261–279). Berlin: Springer.
Kaufmann, L., & Rousseeuw, P. (1990). *Fining groups in data: an introduction to cluster analysis*. New York: Wiley.
Kremer, H., Kranen, P., Jansen, T., Seidl, T., Bifet, A., Holmes, G., & Pfahringer, B. (2011). An effective evaluation measure for clustering on evolving data streams. In *ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 868–876). New York: ACM.
Lam, W. M., & Reibman, A. R. (1993). Design of quantizers for decentralized estimation systems. *IEEE Transactions on Communications*, *41*(11), 1602–1605.
Lu, T., & Boutilier, C. (2011). Learning Mallows models with pairwise preferences. In *International conference on machine learning* (pp. 145–152).
Luce, R. D. (1959). *Individual choice behavior: a theoretical analysis*. New York: Wiley.
Mallows, C. L. (1957). Non-null ranking models. *Biometrika*, *44*, 114–130.
Megalooikonomou, V., & Yesha, Y. (2000). Quantizer design for distributed estimation with communication constraints and unknown observation statistics. *IEEE Transactions on Communications*, *48*(2), 181–184.
Meilă, M., & Bao, L. (2010). An exponential model for infinite rankings. *Journal of Machine Learning Research*, *11*, 3481–3518.
Plackett, R. L. (1975). The analysis of permutations. *Applied Statistics*, *24*(2), 193–202.
Qin, T., Liu, T. Y., Zhang, X., & Li, H. (2008). Global ranking using continuous conditional random fields. In *Advances in neural information processing systems* (pp. 1281–1288). Cambridge: MIT Press.
Qin, T., Geng, X., & Liu, T. Y. (2010). A new probabilistic model for rank aggregation. In *Advances in neural information processing systems* (pp. 1948–1956). Cambridge: MIT Press.
Seo, S., Bode, M., & Obermayer, K. (2003). Soft nearest prototype classification. *IEEE Transactions on Neural Networks*, *14*, 390–398.
Vilalta, R., & Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artificial Intelligence Review*, *18*, 77–95.
Weigend, A. S., Mangeas, M., & Srivastava, A. N. (1995). Nonlinear gated experts for time series: discovering regimes and avoiding overfitting. *International Journal of Neural Systems*, *6*, 373–399.
Yu, P., Wan, W. M., & Lee, P. H. (2010). Decision tree modeling for ranking data. In *Preference learning* (pp. 83–110).
Zhuang, J., & Hoi, S. C. H. (2011). A two-view learning approach for image tag ranking. In *International conference on web search and data mining* (pp. 625–634). New York: ACM.