Supervised Clustering of Label Ranking Data

Mihajlo Grbovic* Nemanja Djuric* Slobodan Vucetic*

Abstract

In this paper we study supervised clustering in the context of label ranking data. Segmentation of such complex data has many potential real-world applications. For example, in target marketing, the goal is to cluster customers in the feature space by taking into consideration the assigned, potentially incomplete product preferences, such that the preferences of instances within a cluster are more similar than the preferences of customers in the other clusters. We establish several heuristic baselines for this application that make use of well-known algorithms such as K-means, and propose a principled algorithm specifically tailored for this type of clustering. It is based on the Plackett-Luce (PL) probabilistic ranking model. Each cluster is represented as a union of Voronoi cells defined by a set of prototypes and is assigned a set of PL label scores that determine the cluster-specific label ranking. The unknown cluster PL parameters and prototype positions are determined using a supervised learning technique. Cluster membership and ranking for a new instance is determined by membership of its nearest prototype. The proposed algorithms were empirically evaluated on synthetic and real-life label ranking data. The PL-based method was superior to the heuristically-based supervised clustering approaches. The proposed PL-based algorithm was also evaluated on the task of label ranking prediction. The results showed that it is highly competitive to the state of the art label ranking algorithms, and that it is particularly accurate on data with partial rankings.

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 $D = \{(\mathbf{x}_n, \pi_n), n = 1, ..., N\}$, 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 data mining and machine learning community and has already been extensively studied [6, 3, 14, 2, 21]. A nice survey of recent label ranking algorithms can be found in [12].

There are many practical applications in which the objective is to learn an exact label preference of an instance in form of a total order. 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.), one might not be interested only in predicting which topics are relevant for a specific document, but also to rank the topics by relevance. Additional applications include: meta-learning [24], 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; predicting food preferences for new customers based on the survey results, demographics, and other characteristics of respondents [17]; determining an order of questions in a survey for a specific user based on respondent's attributes. See [8] for an overview of label ranking applications in economics, operations research, and databases.

Supervised clustering of label ranking data is an open and non-trivial problem, that has not been addressed in the data mining literature. Traditionally, clustering techniques are unsupervised, and as such do not take class memberships (classification) or target values (regression) into consideration. Supervised clustering [9, 10, 11], on the other hand, does. It aims at producing desirable clusterings given the additional information (e.g., class labels). Example applications include clustering news articles by whether they refer to the same topic. Depending on the actual application, there is usually a specific performance measure or reward to evaluate the potential clustering solution.

One can envision many potential applications of su-

*Department of Computer and Information Sciences, Center for Data Analytics and Biomedical Informatics, Temple University, {mihajlo.grbovic, nemanja.djuric, slobodan.vucetic}@temple.edu

pervised clustering for label ranking data. For example, in target marketing, a company with several products would like to cluster its customers for purposes of designing cluster-specific promotional material. 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 a survey for specific target group to maximize its success. For example, given the labeled data of users that did finish the survey, the goal would be to cluster these users into a predetermined number of groups and learn the best question order for each group.

Specifically, in the supervised clustering, the goal is to cluster the data based on both instance features and the assigned, potentially incomplete, label rankings, such that the label ranks of instances within a cluster are more similar to each other than they are to the label ranks of instances in the other clusters. If we were to consider label ranking as classification, where each permutation is treated as a different class, we would have $L!$ classes, and even more when dealing with incomplete ranks. This would make it very hard to apply standard supervised clustering algorithms.

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 also propose a principled probabilistic model. The first baseline approach uses the well-known K-means algorithm, either by treating label rankings as one of the features, or by first clustering based on the instance features only and subsequently assigning label rankings to the obtained clusters using a generalized Mallows Model [18]. The second baseline approach first clusters the label rankings [19], without taking the instance features into account, to obtain a predetermined number of classes. It then trains a multi-class classifier using the newly formed classification data.

The proposed Plackett-Luce Mixture Model (MM-PL) presents a general framework for label ranking that can be used both for supervised clustering and for prediction. It is based on a multi-prototype cluster representation, where the underlying cluster preferences are modeled using cluster-specific Plackett-Luce score parameters. 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.

2 Preliminaries

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 as a permutation π of the set $\{1, ..., L\}$, where L is the number of available class labels. We define π such that $\pi(i)$ is the class label at i-th position in the order and $\pi^{-1}(j)$ is the position of the y_j class label in the order. The permutation can also describe incomplete rankings in form $y_{\pi(1)} \succ_{\mathbf{x}} y_{\pi(2)} \succ_{\mathbf{x}}, ..., \succ_{\mathbf{x}} y_{\pi(l)}$, where $l < L$ and $\{\pi(1), ..., \pi(l)\} \subset \{1, ..., L\}$.

Let us assume that N historical observations are collected in form of a data set $D = \{(\mathbf{x}_n, \pi_n), n = 1, ..., N\}$ and available for development of a label ranking clustering model. The objective is to segment D into K clusters, such that each cluster spans a certain portion of feature space $S_k, k = 1, ..., K$ and is associated with a certain central ranking $\rho_k, k = 1, ..., K$, both to be determined from D. Preferably, we would like $\pi_n, \mathbf{x}_n \in S_k$ to be similar and all at a short distance from ρ_k. The clustering model prediction for a new unlabeled instance $\mathbf{x}_u$ is is given in form of a total order $\hat{\pi}_u = \rho_m$, where m is the index of the cluster that $\mathbf{x}_u$ is assigned to.

To evaluate the quality of potential clustering solutions on previously unseen examples, we will measure how close is the true label rank to the one predicted on the basis of cluster membership. Intuitively, we measure 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.

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

(2.1)
$$d_\tau = |\{(y_i, y_j) : \pi_1^{-1}(y_i) > \pi_1^{-1}(y_j) \wedge \pi_2^{-1}(y_j) > \pi_2^{-1}(y_i)\}|.$$

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

We propose the following measure for comparing different clustering solutions. The label ranking loss on the data set D is defined as

$$loss_{LR} = \frac{1}{N} \sum_{n=1}^{N} \frac{2 \cdot d_\tau(\pi_n, \hat{\pi}_n)}{L \cdot (L-1)}, \tag{2.2}$$

where π_n and $\hat{\pi}_n = \rho_m$ are true and predicted rankings for n-th example, respectively. Note that $loss_{LR}$ can be used to calculate the Kendall's tau coefficient as

$$c_\tau = 1 - 2 \cdot loss_{LR}. \tag{2.3}$$

A very similar measure can be used to evaluate cluster compactness,

$$\Delta_p = \frac{1}{K}\sum_{k=1}^{K}\frac{\sum\limits_{n\in S_k} d_\tau(\pi_n,\rho_k)}{\mid S_k \mid}, \tag{2.4}$$

where ρ_k is the k-th cluster central label ranking.

3 The Plackett-Luce Mixture Model

In this section, we first describe the details involving the probabilistic Plackett-Luce model for ranking of alternatives, and then describe the proposed Plackett-Luce Mixture Model for supervised clustering of label ranking data.

The Plackett-Luce model [20] is an extension of the Bradley-Terry model [1] for comparisons involving three or more alternatives. It is defined by the score vector $\mathbf{v} = (\mathbf{v}(1), \mathbf{v}(2), ..., \mathbf{v}(L)) \in \mathbb{R}_+^L$. The probability of any ranking L labels is expressed in terms of $\mathbf{v}$ as

$$\mathbb{P}(\pi|\mathbf{v}) = \prod_{i=1}^{L}\frac{\mathbf{v}(\pi(i))}{\sum\limits_{j=i}^{L}\mathbf{v}(\pi(j))}. \tag{3.5}$$

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

$$\mathbb{P}(\pi = (c,a,b)|\mathbf{v}) = \frac{\mathbf{v}(c)}{\mathbf{v}(c)+\mathbf{v}(a)+\mathbf{v}(b)}\cdot\frac{\mathbf{v}(a)}{\mathbf{v}(a)+\mathbf{v}(b)}\cdot\frac{\mathbf{v}(b)}{\mathbf{v}(b)}, \tag{3.6}$$

where the first multiplier presents the probability $\mathbb{P}(c$ is ranked $1^{st})$, the second presents the probability $\mathbb{P}(a$ is ranked $2^{nd} \mid c$ is ranked $1^{st})$ and the third presents the probability $\mathbb{P}(b$ is ranked $3^{rd} \mid c$ is ranked $1^{st} \wedge$ a is ranked $2^{nd})$. Assuming the label scores $v(a) = 3, v(b) = 1$ and $v(c) = 5$, for example, would result in $\mathbb{P}(\pi = (c, a, b)|\mathbf{v}) = 0.417$.

The Plackett-Luce mixture model (MM-PL) we propose has the following setup. The feature space is divided into K clusters, where each cluster $S_k, k = 1, ..., K$, is represented as a union of Voronoi cells defined by a set of prototypes $\{\mathbf{m}_p, p = 1, ..., P\}$, where $\mathbf{m}_p$ is a d-dimensional vector in input space. The k-th cluster is assigned a Plackett-Luce label score vector $\mathbf{v}_k$, that defines the cluster central ranking ρ_k.

The resulting model is completely defined by $K \cdot P$ prototypes and K PL vectors, $\{(\{\mathbf{m}_p\}_k, \mathbf{v}_k), k = 1, ..., K, p = 1, ..., P\}$. The optimal prototype positions and label scores, with respect to a desired loss function, are learned in a supervised manner. A new instance is assigned to the cluster of its nearest prototype.

The starting point in the algorithm design is to introduce the probability $\mathbb{P}(p \mid \mathbf{x}_n)$ of assigning n-th 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

$$\mathbb{P}(\mathbf{x}_n) = \sum_{p=1}^{P\cdot K}\mathbb{P}(\mathbf{x}_n \mid p)\cdot\mathbb{P}(p), \tag{3.7}$$

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

$$\mathbb{P}(p \mid \mathbf{x}_n) = \frac{\exp\left(-\|\mathbf{x}_n - \mathbf{m}_p\|^2/2\sigma_p^2\right)}{\sum\limits_{u=1}^{P\cdot K}\exp\left(-\|\mathbf{x}_n - \mathbf{m}_u\|^2/2\sigma_p^2\right)}. \tag{3.8}$$

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 codeword,

$$\mathbb{P}(k \mid \mathbf{x}_n) = \frac{\sum\limits_{p\in S_k}\exp\left(-\|\mathbf{x}_n - \mathbf{m}_p\|^2/2\sigma_p^2\right)}{\sum\limits_{u=1}^{P\cdot K}\exp\left(-\|\mathbf{x}_n - \mathbf{m}_u\|^2/2\sigma_p^2\right)}. \tag{3.9}$$

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

$$\mathbb{P}(\pi \mid \mathbf{x}_n) = \sum_{k=1}^{K}\mathbb{P}(k \mid \mathbf{x}_n)\cdot\mathbb{P}(\pi \mid k), \tag{3.10}$$

where $\mathbb{P}(\pi|k)$ is the probability of ranking π if $\mathbf{x}_n$ is generated by k-th cluster. In our approach, we assume the ranking π corresponds to the PL model, and express

$\mathbb{P}(\pi \mid k)$ as $\mathbb{P}(\pi \mid \mathbf{v}_k)$ defined in (3.5). Based on this model, 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 [25]. The mixture model assumes the conditional independence between $\mathbf{x}_n$ and π given k, $\mathbb{P}(\pi \mid \mathbf{x}_n, k) = \mathbb{P}(\pi \mid k)$. For $\mathbb{P}(k \mid \mathbf{x}_n)$ we assume the distribution from (3.9).

Given the training data set D and assuming the mixture model (3.10), we can find the optimal model parameters $\lambda = \{\{\mathbf{m}_p\}_k, \mathbf{v}_k, k = 1, ..., K, \sigma_p\}$ as the ones that maximize the likelihood,

$$\mathbb{P}(\boldsymbol{\pi} \mid \lambda) = \prod_{n=1}^{N} \sum_{k=1}^{K} \mathbb{P}(k \mid \mathbf{x}_n) \cdot \prod_{i=1}^{L_n} \frac{\mathbf{v}_k(\pi_n(i))}{\sum_{j=i}^{L_n} \mathbf{v}_k(\pi_n(j))}, \tag{3.11}$$

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. However, because of the summation, the log-likelihood function $l(\lambda) = \ln \mathbb{P}(\boldsymbol{\pi} \mid \lambda)$ in its original form is not suitable for optimization. For this reason we resort to a more natural alternative to minimization of $l(\lambda)$, the Expectation-Maximization (EM) algorithm [7]. To map the problem onto EM, we define indicator variables I_{nk} as

$$I_{nk} = \begin{cases} 1, & \text{if } \mathbf{x}_n \text{ is generated by } k\text{-th cluster,} \\ 0, & \text{otherwise,} \end{cases} \tag{3.12}$$

and replace the sum over clusters in (3.11) by a product. After this modification, by taking the logarithm, we end up with a simple double-sum, yielding a more tractable log-likelihood function,

(3.13)

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

In the E-step, the expected values for I_{nk}, denoted as h_{nk}, are estimated assuming that the model parameters λ are known,

(3.14)

$$\mathbf{E-step}:$$

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

In the M-step, the parameters of the model are updated assuming the I_{nk} values are known, and replacing each I_{nk} in (3.14) by its expected value h_{nk}. Note that there exists a strict requirement that $\mathbf{v}_k \in \mathbb{R}_+^L, k = 1, ..., K$ 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 transform the original constrained optimization problem to a new optimization problem, which is unconstrained.

Specifically, we minimize the negative log-likelihood $l(\lambda)$ with respect to $\log(\mathbf{v}_k)$ instead of $\mathbf{v}_k$, where $\partial l(\lambda)/\partial \log(\mathbf{v}_k)$ is simply $\mathbf{v}_k \cdot \partial l(\lambda)/\partial \mathbf{v}_k$ [22]. This converts the problem into the unconstrained optimization, which can now be solved using gradient ascent approach. The summary of the resulting M-step ($v.1$) in the stochastic mode is given as follows,

(3.15)

$$\mathbf{M-step}$$

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

where γ is a learning rate, and $c = 1, ..., L_n$.

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 [23, 13].

Let us denote by p_{nn} the index of the nearest prototype for instance $\mathbf{x}$. As σ_p decreases towards zero, the assignment probability $\mathbb{P}(p_{nn}|\mathbf{x})$ approaches one, and the assignment probabilities of the remaining prototypes approach zero. As a result, $p(\pi|\mathbf{x})$ from (3.10) can be approximated as $p(\pi|\mathbf{x}) \approx p(\pi|k_{nn})$, where k_{nn} is the p_{nn}-th prototype's cluster. Therefore, the label rank for instance $\mathbf{x}$ can be predicted by using the label score vector of its nearest prototype's cluster. Since the optimal ranking for any cluster $\pi_k, k = 1, ..., K$, 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.

4 Heuristic LR Clustering Baselines

We propose the following baseline approaches for label ranking clustering.

1**-Rank** is a simple method used as a true baseline. It shows how appropriate it would be to consider all examples as a single cluster, $K = 1$, and derive a single central ranking ρ_K. Intuitively, this would correspond to designing a single catalog for all customers. The central ranking is found using the probabilistic Mallows model [18].

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 $\hat{d}$,

$$\mathbb{P}(\rho \mid \theta, \pi) = \frac{\exp\left(-\theta \hat{d}(\rho, \pi)\right)}{Z(\theta, \pi)}, \tag{4.16}$$

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

Let us discuss how Mallows model can be used to derive cluster centroids $\{\rho_k\}, k = 1, ..., K$ for K clusters, in general. Then, $K = 1$ is just a special case. Given $|S_k|$ label rankings $\{\pi_n\}, n \in S_k$ for cluster k and assuming independence, the probability that we observe $\boldsymbol{\pi}_k = \{\pi_n\}, n \in S_k$ is

$$\mathbb{P}(\boldsymbol{\pi}_k \mid \theta, \rho_k) = \prod_{n \in S_k} \mathbb{P}(\pi_n \mid \theta, \rho_k). \tag{4.17}$$

The MLE of ρ_k is the one that minimizes

$$loss_{LR}(k) = \frac{1}{|S_k|} \sum_{n \in S_k} \frac{2 \cdot d_\tau(\pi_n, \rho_k)}{L \cdot (L-1)}, \tag{4.18}$$

where the solution can be found by exhaustive 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. To avoid the first issue we make use of the approximate solution that uses a simple Borda count algorithm [5]. Having $|S_k|$ label rankings $\{\pi_n\}, n \in S_k$, 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 $\pi_n, n \in S_k$, 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 [4] that Kendall's tau is well approximated by Spearman's rank correlation, whose median can be computed using Borda.

To solve the second issue we modify Borda count 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 $\{\pi_n^*\}, n \in S_k$. This procedure iterates until ρ_k converges.

K-means $\rightarrow$ Mallows algorithm first performs K-means clustering based on the instance features only, without taking their label rankings into account. It then derives central rankings ρ_k for each cluster from label rankings that belong to this cluster using the Mallows Model. This approach is expected to work well when clusters in the feature space correspond to clusters in the label space.

Naïve K-means In this 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 j^{th} 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.

Table 1: Data sets for label ranking (synthetic (S), semi-synthetic (SS) and real-world)

name	domain	N	d	L	1-Rank $loss_{LR}$	**name**	domain	N	d	L	1-Rank $loss_{LR}$
circles	S	7,000	2	6	.393	checker	S	7,200	2	6	.398
authorship	SS (A)	841	70	4	.269	iris	SS (B)	150	4	3	.451
bodyfat	SS (B)	252	7	7	.450	pendigits	SS (A)	10,992	16	10	.383
calhousing	SS (B)	20,640	4	4	.380	segment	SS (A)	2,310	18	7	.419
cpu-small	SS (B)	8,192	6	5	.431	stock	SS (B)	950	5	5	.445
elevators	SS (B)	16,599	9	9	.435	vehicle	SS (A)	846	18	4	.335
fried	SS (B)	40,769	9	5	.493	vowel	SS (A)	528	10	11	.334
glass	SS (A)	214	9	6	.171	wine	SS (A)	178	13	3	.234
housing	SS (B)	506	6	6	.412	wisconsin	SS (B)	194	16	16	.485
cold	biology	2,465	24	4	.401	heat	biology	2,465	24	6	.464
diau	biology	2,465	24	7	.348	spo	biology	2,465	24	11	.441
dtt	biology	2,465	24	4	.438	sushi	food	5,000	11	10	.392

Unsupervised Label Ranking → SVM first clusters the label rankings using the unsupervised label ranking approach to obtain K classes (cluster central rankings). It then trains a multi-class SVM classifier with RBF kernel using the newly labeled data. When a new instance is classified, it is assigned a ranking that corresponds to the predicted class. We consider two unsupervised label ranking methods as 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 dataset. 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's 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 the top K ranks really correspond to cluster central rankings. However, if the cluster 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 [19]. 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.

This gave rise to two methods, namely **Naïve → SVM** and **EBMS → SVM**.

5 Experiments

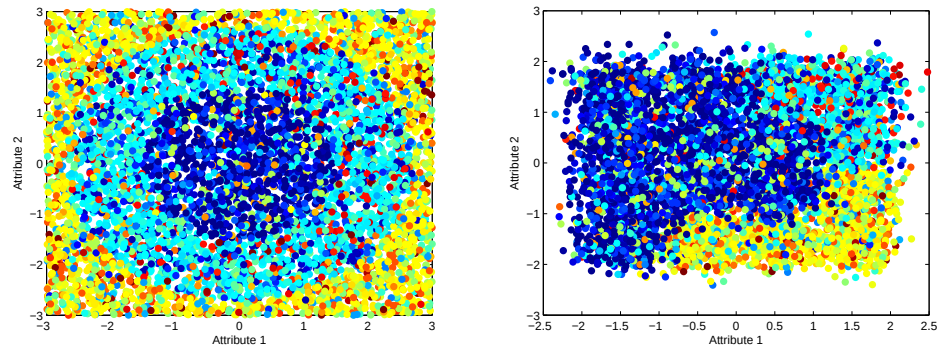
In this section we evaluate the MM-PL algorithm in both supervised clustering and label ranking scenarios. In the clustering scenario, it was compared to the five baselines described in the previous section. In the traditional label ranking scenario, it was compared to several previously proposed algorithms.

In our preliminary experiments, we experimented with several different versions of the MM-PL algorithm. We were interested in comparing batch vs. stochastic prototype updates in the M-step. The main conclusions are that the stochastic updates performed better than batch updates. Following these results, in the following we report the performance of MM-PL with stochastic updates.

Let us first discuss MM-PL implementation details, specifically, initialization and parameter selection. We initialize the prototypes by randomly 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. 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 [23, 13] and updated using the annealing schedule with $\sigma_T = 8N$. Training was terminated when $l(\lambda)$ defined in (3.14) stopped decreasing by more than 10^{-5}.

The data sets are described in Section 5.1. Clustering results are discussed in Section 5.2. Finally, the results in the label ranking application, together with a brief related work overview, are shown in Section 5.3.

5.1 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 [2, 3] 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



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

Figure 1: Synthetic data sets (data point color represents lexicographic order of label ranking)

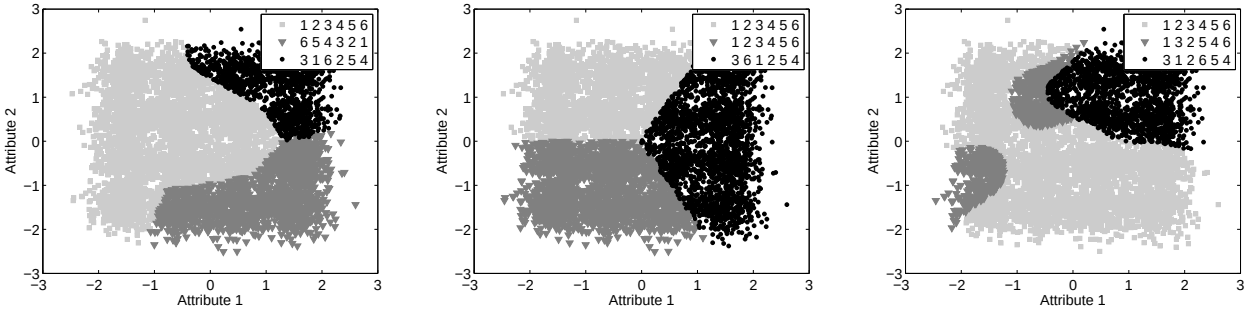
Statlog repositories into label ranking data, by using the following Naïve Bayes (A) and feature-to-label (B) [3] techniques. For multi-class data sets, the Naïve Bayes classifier is first trained on the original data. Then, the label ranking data is obtained by sorting labels with respect to the predicted 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. Since there exists a certain correlation between all the attributes, it will be interesting to see to what extent can the label ranks be correctly predicted.

In addition, we used the real-world label ranking data from [15] and sushi preference data from [17]. The data from the first source includes five data sets from the medical domain. The Yeast genome consists of 2465 genes, each described by the phylogenetic profile of length 24. The 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 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 experiments the data version from [17] was used. The goal is to predict how would a new customer rank 10 sushis based on his/her features. The additional features which describe each sushi dish were excluded since they could not be incorporated in our model.

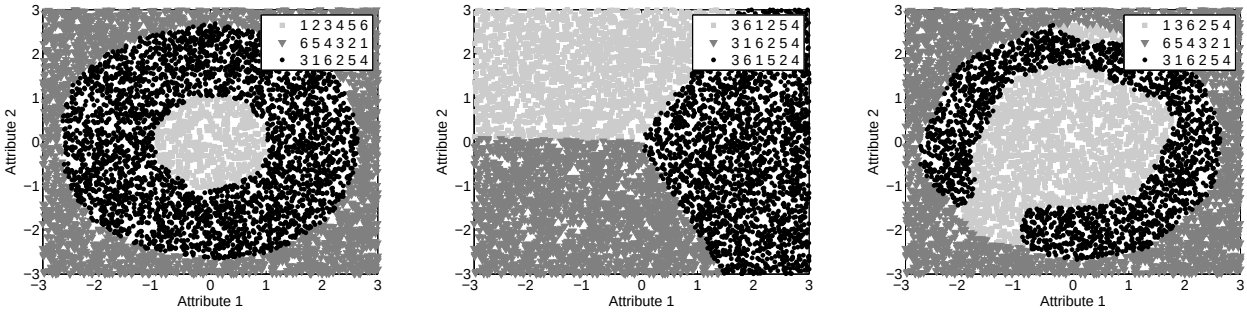
We also generated two 2-dimensional data sets to better characterize our model. These synthetic data sets were separated into 3 classes, where each class has unique label ranking with $L = 6$ labels ($1 : 123456, 2 : 654321, 3 : 316254$). Then, each point was assigned its class' label ranking corrupted 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 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.

Two data sets are shown in Figure 1, where each label permutation is marked with a different color. The



(a) $\underset{PL}{\mathbf{MM}}$, $\Delta_p = .526$ (b) $\underset{Mal}{\mathbf{Km}}$, $\Delta_p = .771$ (c) $\underset{SVM}{\mathbf{ebms}}$, $\Delta_p = .617$

Figure 2: Clustering performance comparison on *checker* dataset ($K = 3$)



(a) $\underset{PL}{\mathbf{MM}}$, $\Delta_p = .532$ (b) $\underset{Mal}{\mathbf{Km}}$, $\Delta_p = .831$ (c) $\underset{SVM}{\mathbf{naive}}$, $\Delta_{=}.552$

Figure 3: Clustering performance comparison on *circle* dataset ($K = 3$)

description of data sets is given in Table 1, together with a 1-Rank result as it is, in a sense, a property of the data set (i.e., how well it can be represented using a single label rank).

A common shortcoming of a 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 (e.g., clothing brands). To model incomplete ranking information, the training data was modified such that a certain percentage of labels was removed from the complete instance rankings at random.

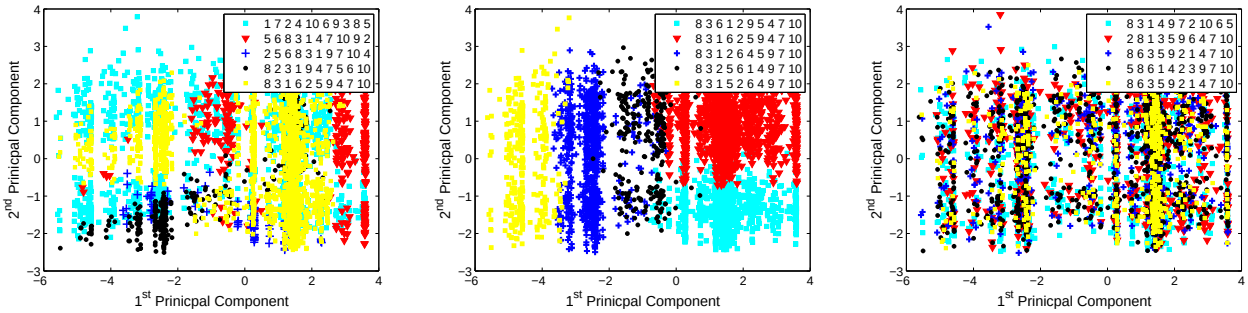
5.2 Application to Supervised Clustering Here we compare MM-PL to clustering algorithms from Section 4 in label rank clustering. The comparison is in terms of $loss_{LR}$ and Δ_p averaged over 5 repetitions of a 10-fold cross-validation. Note that $loss_{LR}$ is used to evaluate predictiveness, while Δ_p depicts cluster compactness, using the training data, which is not necessarily a good measure of predictive performance. Missing label scenario was simulated by removing a certain percentage of labels from the training data.

First we show the results on the synthetic *checker* and *circle* data sets. 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. The results are summarized in the top two rows of Table 4 and illustrated in Figures 2 and 3. In addition, Table 2 reports which rankings are discovered as cluster centroids in different settings.

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 are missing. It can be observed that K-means $\rightarrow$ Mal is under-performing when the label regions have complicated distributions. Naïve $\rightarrow$ SVM top K central ranking selection strategy did not prove efficient. Because of imbalance in cluster sizes and the large amount of noise, it was not able to correctly recover all the central rankings by taking the top K ones. Interestingly, nor did EBMS $\rightarrow$ SVM. Finally, Naïve K-means showed better

Table 2: The resulting cluster central rankings (upper half, checker data, $K = 3$; lower half, circle data, $K = 3$)

complete ranking					30% missing labels					60% missing labels				
Km $_{Mal}$	**naive** $_{Km}$	**naive** $_{SVM}$	**ebms** $_{SVM}$	**MM** $_{PL}$	**Km** $_{Mal}$	**naive** $_{Km}$	**naive** $_{SVM}$	**ebms** $_{SVM}$	**MM** $_{PL}$	**Km** $_{Mal}$	**naive** $_{Km}$	**naive** $_{SVM}$	**ebms** $_{SVM}$	**MM** $_{PL}$
123456	**123456**	**123456**	132654	**123456**	**123456**	**123456**	**123456**	132546	**123456**	**123456**	**123456**	**123456**	132645	**123456**
123456	**654321**	**654321**	653421	**654321**	123456	**654321**	**654321**	132645	**654321**	316254	**654321**	124356	132645	**654321**
361524	423516	213456	653421	**316254**	361254	136254	124356	132645	**316254**	316452	123645	123546	132546	**316254**
361254	**123456**	136254	**123456**	**123456**	361524	132654	312654	653421	**123456**	361524	635124	316524	316254	**123456**
316254	**654321**	**654321**	132546	**654321**	361524	**654321**	**654321**	653421	**654321**	361254	316254	361254	316254	**654321**
361524	**316254**	**316254**	312654	**316254**	361524	654321	**316254**	132654	**316254**	361524	**316254**	**316254**	**316254**	**316254**



(a) $\underset{PL}{\mathbf{MM}}$, $\Delta_p = .641$ (b) $\underset{Mal}{\mathbf{Km}}$, $\Delta_p = .695$ (c) $\underset{SVM}{\mathbf{ebms}}$, $\Delta_p = .645$

Figure 4: Clustering performance comparison on *sushi* dataset ($K = 5$)

Table 3: The resulting cluster central rankings for Sushi data set $K = 5$, no missing labels

1 **Rank**		**Km** $_{Mal}$	**naive** $_{Km}$	**naive** $_{SVM}$	**ebms** $_{SVM}$	**MM** $_{PL}$
		8 3 6 1 2 9 5 4 7 10	5 2 1 8 6 4 3 9 7 10	2 5 6 8 3 1 9 7 10 4	8 3 1 4 9 7 2 10 6 5	1 7 2 4 10 6 9 3 8 5
		8 3 1 6 2 5 9 4 7 10	1 7 4 2 3 10 9 8 6 5	5 6 8 3 1 4 9 2 7 10	2 8 1 3 5 9 6 4 7 10	5 6 8 3 1 4 7 10 9 2
8 3 1 6 2 5 9 4 7 10		8 3 1 2 6 4 5 9 7 10	8 3 9 1 2 6 4 7 10 5	8 3 6 5 9 2 1 4 10 7	8 6 3 5 9 2 1 4 7 10	2 5 6 8 3 1 9 7 10 4
		8 3 2 5 6 1 4 9 7 10	8 5 2 6 3 9 1 7 4 10	8 5 6 2 1 3 9 4 10 7	5 8 6 1 4 2 3 9 7 10	8 2 3 1 9 4 7 5 6 10
		8 3 1 5 2 6 4 9 7 10	8 5 3 6 1 4 9 2 7 10	1 8 3 9 2 4 7 10 6 5	8 6 3 5 9 2 1 4 7 10	8 3 1 6 2 5 9 4 7 10
Δ_p	.695	.778	.652	.645	**.641**	

performance than K-means → Mal on the *checker* data but falls behind it on the *circle* data.

Table 2 shows that none of the algorithms, except MM-PL, was able to find the correct label ranking for the smallest *checker* cluster.

It should be noted that, if we repeat the experiment multiple times, K-means based algorithms are likely to find different central rankings each time. On the other hand, the other algorithms are consistent.

Next, we evaluate the algorithms on the real-world sushi data set with $K = 5$ clusters. Figure 4, Table 3 and bottom row of Table 4 illustrate the performance. We can make several observations. Even though K-means → Mal partitions the feature space well, the partitions are not descriptive enough, as all cluster central rankings have 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 the in mid-part of customer preferences. EBMS → SVM results in the partitions that are compact in the label space (low Δ_p), but far less compact in the feature space, thus resulting in poor generalization (Table 4). MM-PL clusters are informative and diverse in label space and also consistently distributed in feature space, allowing for a simpler and more intuitive rule when assigning new customers to clusters. For this reason it has the best overall predictive performance.

The results on the remaining datasets are reported in Table 4. We show results for complete ranking, 30% and 60% missing label ranking scenarios. The number of clusters was set to $K = 10$ and the total budget size of MM-PL to $P \cdot K = 100$.

MM-PL performed the best overall in all three scenarios. MM-PL is the best overall, while K-means → Mal and Naïve → SVM are the second best, depending on the data properties. If the clusters in feature space correspond to clusters in the label space, K-means → Mal performs well. If top K label rankings correspond to the true cluster centroids, then Naïve → SVM is a good fit.

Table 4: Label Ranking Supervised Clustering performance comparison in terms of $loss_{LR}$

	complete ranking					30% **missing labels**					60% **missing labels**				
Data Set	**Km** $_{Mal}$	**naive** $_{Km}$	**naive** $_{SVM}$	**ebms** $_{SVM}$	**MM** $_{PL}$	**Km** $_{Mal}$	**naive** $_{Km}$	**naive** $_{SVM}$	**ebms** $_{SVM}$	**MM** $_{PL}$	**Km** $_{Mal}$	**naive** $_{Km}$	**naive** $_{SVM}$	**ebms** $_{SVM}$	**MM** $_{PL}$
checker	.340	.287	.287	.373	**.250**	.340	.300	.287	.377	**.249**	.342	.308	.413	.393	**.260**
circle	.392	.444	.260	.281	**.253**	.392	.410	.265	.280	**.258**	.392	.396	.374	.395	**.261**
fried	.413	.334	.272	.263	**.226**	.412	.342	.291	.340	**.236**	.422	.395	.317	.375	**.251**
calhousing	.364	.424	.450	.359	**.349**	.365	.463	.467	.379	**.352**	.365	.501	.469	.390	**.349**
elevators	.347	.300	.220	.182	**.171**	.334	.301	.272	.248	**.210**	.336	.308	.293	.271	**.214**
pendigits	.290	.308	**.148**	.314	.172	.289	.287	.197	.326	**.181**	.291	.337	.236	.349	**.202**
cpu-small	.362	.374	.407	.403	**.356**	.363	.389	.390	.399	**.356**	.365	.413	.407	.387	**.360**
segment	.214	.188	**.080**	.185	.146	.214	.197	**.115**	.184	.145	.213	.229	.202	.226	**.156**
wisconsin	.451	.443	.486	.453	.433	.455	.443	.487	.462	.435	.465	.461	.490	.469	**.444**
vowel	.253	.274	.237	.221	.209	.255	.276	.266	.239	**.211**	.265	.314	.305	.256	**.242**
vehicle	.132	.113	.095	.124	.090	.131	.165	.080	.155	**.095**	.136	.210	.142	.174	**.109**
stock	.213	.218	**.138**	.235	.140	.217	.227	.175	.241	**.148**	.222	.259	.273	.310	**.162**
iris	.091	.059	.080	.171	.061	.093	.142	.100	.174	**.088**	.136	.188	.196	.256	**.127**
glass	.110	.113	.092	.112	.108	.117	.173	**.098**	.125	.114	.139	.211	.189	.144	**.130**
authorship	.072	.066	.101	.175	.079	.073	**.072**	.109	.179	.075	.082	**.077**	.193	.221	.082
bodyfat	.451	.454	.461	**.419**	.430	.455	.459	.464	.453	.436	.469	.464	.491	.458	**.444**
wine	.050	.058	.051	.106	**.047**	.051	.163	.059	.122	**.045**	.087	.220	.104	.133	**.076**
housing	.376	.407	**.331**	.366	.339	.377	.421	.381	.412	**.357**	.397	.436	.384	.426	**.365**
cold	.395	.408	.416	.426	**.386**	.405	.404	.448	.437	**.393**	.404	.418	.434	.438	**.398**
heat	.438	.453	.477	.465	**.433**	.440	.451	.495	.472	**.437**	.447	.456	.477	.485	**.439**
diau	.336	.385	.379	.356	**.327**	.339	.376	.408	.366	**.338**	.345	.378	.362	.382	**.343**
dtt	.416	.432	.471	.447	**.410**	.422	.428	.464	.449	**.417**	.424	.432	.466	.453	**.422**
spo	**.431**	.440	.484	.463	**.431**	**.434**	.445	.487	.474	.435	.443	.471	.481	.478	**.440**
sushi	.372	.383	.379	.366	**.332**	.369	.374	.414	.372	**.335**	.379	.394	.382	.477	**.343**
average	.305	.307	.284	.303	**.257**	.306	.321	.301	.319	**.264**	.315	.345	.336	.348	**.276**
time (s)	10.1	9.3	1300	13K	234	13.5	10.9	3190	13.2K	228	13.8	11.6	3730	13.3K	188

Finally, we can observe that the Unsupervised Label Ranking → SVM methods are notably computationally more demanding than the rest of the algorithms. Especially EBMS, which is very slow on large data sets.

5.3 Application to Label Ranking Prediction

The objective of label ranking is to train a single ranking function $f : \mathbf{x}_n \rightarrow \pi_n$ from data set D. As this function can output any permutation of L labels, it is not appropriate for clustering (e.g., it could result in an infeasible number of catalogs for target marketing). However, many other real-life tasks find this kind of mapping desirable.

There are three principal approaches for label ranking. The first decomposes label ranking problem into one or several binary classification problems. Ranking by pairwise comparison (PW) [15], for example, creates $L \cdot (L-1)/2$ classification problems, one for each possible pairwise ranking. The constraint classification (CC) [14], on the other hand, transforms the label ranking problem into a single binary classification problem by augmenting the data.

The second approach is to use utility functions, where the goal is to learn mappings $f_k : X \rightarrow R$ for each label $k = 1, ..., L$, which assign a value $f_k(\mathbf{x})$ to each label, such that $f_i(\mathbf{x}) < f_j(\mathbf{x})$ if $\mathbf{x}$ prefers label j over i. For example, Lin-LL method proposed in [6] represents each f_k as a linear combination of base ranking functions. The utility functions are learned to minimize the ranking error and the final rank is produced by sorting the utility scores.

The third approach is represented by a collection of algorithms which use probabilistic approaches for label ranking, such as the Mallows [18] and the Plackett-Luce [20] models. A typical representative is the instance-based (IB) label ranking [3, 2]. Given a new instance $\mathbf{x}$, the k-Nearest Neighbor algorithm is used to locate its neighbors in the feature space. Then, the neighbors' label rankings are aggregated to provide prediction. Instance models based on the Mallows model, IB-Mal [3] and Plackett-Luce model, IB-PL [2] exist. By setting $k = 1$, the aggregation step is avoided and prediction consists of simply copying the ranking of the nearest neighbor. However, the 1-nearest neighbor is rather unstable and not applicable to incomplete label ranks.

Instance-based label ranking algorithms are simple and intuitive. Furthermore, they have shown good performance in various label ranking scenarios. However, their success comes at a large cost associated with both memory and time. First, they require

Table 5: Label Ranking Prediction performance comparison in terms of label rank loss $loss_{LR}$

	complete ranking							30% missing labels					60% missing labels				
Data Set	1**NN**	**IB** $_{CPS}$	**IB** $_{PL}$	**IB** $_{Mal}$	**Lin** $_{LL}$	**Lin** $_{PL}$	**MM** $_{PL}$	**IB** $_{PL}$	**IB** $_{Mal}$	**Lin** $_{LL}$	**Lin** $_{PL}$	**MM** $_{PL}$	**IB** $_{PL}$	**IB** $_{Mal}$	**Lin** $_{LL}$	**Lin** $_{PL}$	**MM** $_{PL}$
fried	.248	.201	.202	**.196**	.344	.347	.294	**.216**	.219	.345	.348	.300	**.257**	.312	.349	.350	.313
calhousing	.386	.344	.337	**.336**	.449	.435	.351	**.345**	.348	.450	.436	.353	**.376**	.415	.453	.437	.356
elevators	.261	**.224**	.234	.226	.306	.305	.244	**.238**	.244	.309	.306	.249	**.263**	.322	.311	.307	.265
pendigits	**.142**	.140	.143	.139	.364	.351	.211	**.152**	.159	.366	.354	.215	**.169**	.230	.368	.355	.219
cpu-small	.392	.347	.344	**.336**	.378	.361	.348	.351	**.350**	.379	.369	**.350**	.371	.408	.380	.370	**.355**
segment	**.073**	.077	.089	.089	.241	.227	.132	**.111**	.117	.242	.228	.135	**.124**	.154	.242	.226	.149
wisconsin	.458	.492	.435	.419	.403	**.401**	.413	.434	.427	.409	**.403**	.418	.445	.443	.422	**.412**	.430
vowel	**.110**	.117	.158	.148	.313	.317	.169	.217	.178	.316	.318	**.173**	.234	.240	.329	.327	**.231**
vehicle	.097	.087	.082	**.079**	.126	.115	.083	**.093**	.095	.129	.118	**.093**	.113	.131	.128	.125	**.102**
stock	.107	.118	.106	.108	.283	.268	**.105**	.134	.133	.284	.269	**.130**	.154	.169	.285	.272	**.151**
iris	.043	.049	.040	**.037**	.189	.171	.047	**.059**	.066	.199	.172	.064	**.103**	.118	.208	.174	.105
glass	**.075**	.102	.098	.094	.107	.106	.090	.107	.111	.111	.108	**.103**	**.115**	.140	.124	.118	.120
authorship	.080	.063	.063	**.062**	.182	.175	**.062**	.068	.072	.189	.178	**.065**	.085	.108	.200	.190	**.081**
bodyfat	.472	.436	.442	.438	.422	**.404**	.423	.463	.448	.430	**.406**	.430	.466	.460	.431	**.429**	.439
wine	.068	.039	.040	.036	.067	.059	**.034**	.041	.042	.071	.064	**.038**	.061	.077	.101	.079	**.059**
housing	.323	**.321**	.329	.334	.398	.385	.330	.355	.357	.404	.386	**.348**	.384	.385	.408	.391	**.364**
cold	.425	.389	.386	.387	.409	.399	**.385**	.398	.398	.410	.404	**.387**	.424	.423	.411	.405	**.397**
heat	.446	.434	.438	.435	.439	.438	**.430**	.445	.433	.441	.442	**.432**	.460	.455	.445	.443	**.435**
diau	.382	.336	.336	.332	.350	.349	**.329**	.338	.340	.353	.350	**.332**	.339	.365	.360	.352	**.338**
dtt	.440	.420	.418	.417	.421	.410	**.410**	.428	.430	.422	.417	**.414**	.447	.442	.423	**.419**	.423
spo	.440	.435	.438	.433	.437	.443	**.428**	.439	.444	.439	.445	**.434**	.441	.455	.444	.453	**.436**
sushi	.415	.347	.348	.349	.459	.462	**.337**	.363	.352	.461	.465	**.339**	.381	.374	.463	.468	**.342**
average	.267	.253	.250	**.247**	.322	.315	.255	**.263**	.264	.325	.318	**.263**	.282	.301	.331	.323	**.277**
avg. rank	4.73	3.23	3.41	**2.55**	6	5.27	2.82	2.36	2.77	4.32	3.77	**1.77**	2.45	3.64	3.95	3.14	**1.82**

that the entire training data set is stored in memory, which can be costly or even impossible in the resource-constrained applications. Storing the original data can also raise privacy issues, as the data might contain sensitive user information. Second, the prediction involves costly nearest neighbor search and aggregation of neighbors' label rankings. The aggregation is very slow as it requires using optimization techniques at prediction time, such as Minorization-Maximization [16] (IB-PL, $\mathcal{O}(N^2 + kL \log L)$) or exhaustive search (IB-Mal, $\mathcal{O}(N^2 + kL!)$).

There also exists a method that models the global Plackett-Luce parameters which are linearly dependent on the input space [2]. We will refer to it as Lin-PL.

In the following we compare the proposed MM-PL algorithm to simple 1-NN label rank rule, IB-PL [2], IB-Mal [3], Lin-LL [6] and Lin-PL [2]. We also implemented an IB-CPS method which uses the recently proposed CPS model for rank aggregation [21].

Table 5 shows the comparison of MM-PL to the label ranking algorithms. Since the goal in this application is to predict the label preferences as accurately as possible, we used a special case of MM-PL. We set the number of clusters equal to total number of prototypes, hoping to span a much larger number of preferences and increase our predictive power. However, not all prototypes will necessarily learn to represent different preferences. In these experiments we set $K = 100$.

In the full ranking scenario, MM-PL was the second best ranked algorithm in terms of the number of wins. By the average accuracy measure, it was slightly less accurate than the two IB algorithms (IB-Mal and IB-PL), and as accurate as IB-CPS. Lin-LL and Lin-PL were the least accurate by both measures. Accuracy of 1-NN was between Lin and IB methods. It is worth observing that MM-PL was the most accurate on the real-world data sets and that in some cases, due to overfitting, MM-PL with 10 clusters outperforms the new version.

In the case of 30% and 60% missing label information, MM-PL was the highest ranked algorithm. In average accuracy, MM-PL and IB-PL performed similarly. Better performance of MM-PL in the missing label scenario can be explained by the difference in utilizing the incomplete preference information. Unlike IB methods, which use missing labels in the prediction phase, MM-PL prototypes capture the underlying distribution better and alleviate missing data problem by combining partial information during the training phase. It is also interesting to observe that IB-PL performs better than IB-Mal on data with missing labels, while on complete ranking data the outcome is opposite.

6 Conclusion

In spite of having many potential real-world applications, to the best of our knowledge, this paper presents the first attempt at supervised clustering of complex label rank data. We established several baselines for supervised clustering of label ranking data and proposed a Plackett-Luce (PL) mixture model specifically tailored for this application. We empirically showed the strength of the PL model by experiments on real-world and syn-

thetic data. In addition to the supervised clustering scenario, we compared the PL model to the previously proposed label ranking algorithms in terms of predictive accuracy.

Our model achieved state-of-the-art clustering performance and was very competitive to the previously proposed approaches for label ranking prediction. When faced with the missing label problem, and particularly when a large fraction of labels is missing, the PL model works exceptionally well. In addition to the impressive accuracy, the PL model has a small memory footprint making it attractive in memory-constrained scenarios. Unlike models utilizing pairwise label preference information, training and prediction time of PL model are only linear in the size of label set and number of prototypes, resulting in a very fast and efficient model. These features make it an attractive option for label ranking.

Acknowledgments

We are grateful to Marina Meila for providing the codes for the EBMS algorithm.

References

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

[2] W. Cheng, K. Dembczyński, and E. Hüllermeier, *Label ranking methods based on the Plackett-Luce model*, in IEEE International Conference on Machine Learning (ICML), 2010, pp. 215–222.

[3] W. Cheng, J. Hühn, and E. Hüllermeier, *Decision tree and instance-based learning for label ranking*, in IEEE International Conference on Machine Learning (ICML), 2009, pp. 161–168.

[4] D. Coppersmith, L. K. Fleischer, and A. Rudra, *Ordering by weighted number of wins gives a good ranking of weighted tournaments*, ACM-SIAM Symposium on Discrete Algorithms, (2006), pp. 776–782.

[5] J. C. de Borda, *Memoire sur les Élections au Scrutin*, Histoire de l'Academie Royale des Sciences, 1781.

[6] O. Dekel, C. Manning, and Y. Singer, *Log-Linear Models for Label Ranking*, in Advances in Neural Information Processing Systems, vol. 16, MIT Press, 2003.

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

[8] C. Domshlak, E. Hüllermeier, S. Kaci, and H. Prade, *Preferences in AI: An overview*, Artificial Intelligence, 175 (2011), pp. 1037–1052.

[9] C. Eick, N. Zeidat, and Z. Zhao, *Supervised Clustering - Algorithms and Benefits*, Proc. International Conference on Tools with AI, (2011), pp. 774–776.

[10] C. F. Eick, B. Vaezian, D. Jiang, and J. Wang, *Discovery of Interesting Regions in Spatial Data Sets*, European Conference on Principles and Practice of Knowledge Discovery in Databases, (2011).

[11] T. Finley and T. Joachims, *Supervised Clustering with Support Vector Machines*, International Conference, (2005), p. 217224.

[12] T. Gärtner and S. Vembu, *Label Ranking Algorithms: A Survey*, in Preference Learning, E. H. Johannes Fürnkranz, ed., Springer–Verlag, 2010.

[13] M. Grbovic and S. Vucetic, *Regression Learning Vector Quantization*, IEEE International Conference on Data Mining (ICDM), (2009), pp. 788–793.

[14] S. Har-Peled, D. Roth, and D. Zimak, *Constraint classification for multiclass classification and ranking*, in Proceedings of the 16th Annual Conference on Neural Information Processing Systems, NIPS-02, MIT Press, 2003, pp. 785–792.

[15] E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Brinker, *Label ranking by learning pairwise preferences*, Artificial Intelligence, 172 (2008), pp. 1897–1916.

[16] D. R. Hunter, *MM algorithms for generalized Bradley-Terry models*, The Annals of Statistics, 32 (2004), pp. 384–406.

[17] T. Kamishima and S. Akaho, *Filling-in Missing Objects in Orders*, in IEEE International Conference on Data Mining (ICDM), IEEE Computer Society, 2004, pp. 423–426.

[18] C. L. Mallows, *Non-null ranking models*, Biometrika, 44 (1967), pp. 114–130.

[19] M. Meila and L. Bao, *An Exponential Model for Infinite Rankings*, Journal of Machine Learning Research, 11 (2010), pp. 3481–3518.

[20] R. L. Plackett, *The analysis of permutations*, Applied Statistics, 24 (1975), pp. 193–202.

[21] T. Qin, X. Geng, and T.-Y. Liu, *A New Probabilistic Model for Rank Aggregation*, in Advances in Neural Information Processing Systems, MIT Press, 2010, pp. 1948–1956.

[22] T. Qin, T.-Y. Liu, X. Zhang, and H. Li, *Global Ranking Using Continuous Conditional Random Fields*, in Advances in Neural Information Processing Systems, MIT Press, 2008, pp. 1281–1288.

[23] S. Seo, M. Bode, and K. Obermayer, *Soft nearest prototype classification*, IEEE Transactions on Neural Networks, 14 (2003), pp. 390–398.

[24] R. Vilalta and Y. Drissi, *A Perspective View and Survey of Meta-Learning*, Artificial Intelligence Review, 18 (2002), p. 7795.

[25] A. S. Weigend, M. Mangeas, and A. N. Srivastava, *Nonlinear Gated Experts for Time Series: Discovering Regimes and Avoiding Overfitting*, International Journal of Neural Systems, 6 (1995), pp. 373 –399.