Learning from Pairwise Preference Data using Gaussian Mixture Model

Mihajlo Grbovic and Nemanja Djuric and Slobodan Vucetic

Abstract. In this paper we propose a fast online preference learning algorithm capable of utilizing incomplete preference information. It is based on a Gaussian mixture model that learns soft pairwise label preferences via minimization of the proposed soft rank loss measure. Standard supervised learning techniques, such as gradient descent or Expectation Maximization can be used to find the unknown model parameters. Algorithm outputs are soft pairwise label preference predictions that need to be further aggregated to produce a total label ranking prediction, for which several existing algorithms can be used. The main advantages of the proposed learning algorithm are the ability to process a single training instance at a time, low time and space complexity, ease of implementation, and model reuse.

1 INTRODUCTION

Label Ranking is emerging as an important and practically relevant preference learning field. Unlike the standard problems of classification and regression, label ranking learning is a complex learning task, which involves the 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 \( x \), is assigned a ranking of labels \( \pi \), that is a total (e.g. \( \pi = (3, 5, 1, 4, 2) \)) 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 \( x \) to a total label order \( h : x_n \rightarrow \pi_n \). It is assumed that a sample from the underlying distribution \( D = \{(x_n, \pi_n), n = 1, \ldots, N\} \), where \( x_n \) is a \( d \)-dimensional feature vector and \( \pi_n \) 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 [6, 4, 9, 3, 16]. A survey of recent label ranking algorithms can be found in [8].

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 [18], 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 [12]; determining an order of questions in a survey for a specific user based on respondent’s attributes. A recent publication [14], suggests clustering of label ranking data, which could be of great practical importance, especially in target marketing.

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) [11], for example, creates \( L \cdot (L − 1)/2 \) classification problems, one for each possible pairwise ranking. Pairwise binary classifier predictions are aggregated into a total label order by voting. The constraint classification (CC) [9], on the other hand, transforms the label ranking problem into a single binary classification problem by augmenting the data set, such that each example \( x \) is mapped into \( L \cdot (L − 1)/2, (d \times L) \)-dimensional, examples. This allows for training of a single classifier.

The second approach is to use utility functions, where the goal is to learn mappings \( f_k : X \rightarrow R \) for each label \( k = 1, \ldots, L \), which assign a value \( f_k(x) \) to each label, such that \( f_k(x) < f_l(x) \) if \( x \) prefers label \( j \) over \( i \). For example, the label ranking method proposed in [6] represents each \( f_k \) as a linear combination of base ranking functions. The utility functions are learned to minimize the number of ranking errors and the final rank is produced simply by ranking the utility scores. It should be noted that the utility function-based approach is also popular in the related object ranking problem, where techniques based on SVM [10] and AdaBoost [7] have been proposed.

The third approach is represented by a collection of algorithms which use probabilistic approaches for label ranking, such as the ones that rely on the Mallows [13] and the Plackett-Luce (PL) [15] models. A typical representative is the instance-based (IB) label ranking [4, 3]. Given a new instance \( x \), 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. Rank aggregation for prediction is not a trivial task, particularly in presence of partial label ranks. Mallows [4] and Plackett-Luce [3] models that describe probability distribution of rankings have been used to come up with the optimization criterion for rank aggregation. As an alternative, CPS probabilistic model for rank aggregation has been recently proposed [16].

Instance-based label ranking algorithms are simple and intuitive. Furthermore, they have been shown to outperform the competitors in various label ranking scenarios. However, their success comes at a large cost associated with both memory and time. First, they require 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 slow as it requires using op-
timization techniques at prediction time, such as iterative Minimization
Maximization (IB-PL) or exhaustive search (IB Mallows).

In this paper we propose an online, time- and memory-efficient al-
gorithm for learning label preferences based on the Gaussian Mixture
Model (GMM), which could be attractive because of an intuitively
clear learning process and ease of implementation. The model pre-
serves privacy as it consists of mixtures defined by prototypes which
are not the actual data points. Every prototype is associated with
preference judgments for each pair of labels. Unlike many competi-
tors, our algorithm is not limited to a specific type of label ranking
and could support various ranking structures (bipartite, multipartite,
etc.). For an unlabeled instance, GMM predicts the soft label preferences
by averaging prototypes’ pairwise preferences according to distances.

These soft label preferences in form of a preference matrix need to
be aggregated further, into a total order of labels. This is a well known
problem in preference learning and is an especially popular research
area in the object ranking scenario, where numerous methods have been
proposed [5, 1, 2].

2 PRELIMINARIES

In the label ranking scenario, a single instance, described by a
dimensional vector \( x \), is associated with a total order of assigned
class labels, represented as a permutation \( \pi \).

In the label ranking scenario, a single instance, described by a
dimensional vector \( x \), is associated with a total order of assigned
class labels, represented as a permutation \( \pi \).

In our approach, instead of the total order, we use a zero-diagonal
permutation preference matrix \( Y \). When a preference between labels \( y_i \) and \( y_j \) exists, we set \( Y(i,j) > Y(j,i) \) if \( y_i \) is preferred over \( y_j \) and \( Y(i,j) < Y(j,i) \) otherwise. A value of \( Y(i,j) = 1 \) for \( i, j \in \{1, ..., L\} \). A value of \( Y(i,j) \) which is close to 1 is interpreted as
a strong preference that \( y_i \) should be ranked above \( y_j \). A typical
approach is to assign \( Y(i,j) = 1 \) and \( Y(j,i) = 0 \) if \( y_i \) is preferred over \( y_j \). Similarly, uncertain (soft) preferences can be modeled by using
values lower than 1. For example, indifferences (ties) are represented by
setting \( Y(i,j) = 1 \) and \( Y(j,i) = 0.5 \) in case of non-existing, incompar-
able or missing preferences, both \( Y(i,j) = 0 \) and \( Y(j,i) = 0 \).

This representation allows us to work with complete and partial
label orders, as well as with pairwise preferences with uncertainties
and indifferences. Finally, bipartite and multi-partite label rankings
could be handled as well.

Evaluation metrics. Let us assume that \( N \) historical observations
are collected in a form of a data set \( D = \{(x_n, Y_n) \}, n = 1, ..., N \).
The objective in all scenarios is to train a ranking function \( h : x_n \rightarrow \hat{\pi}_n \) from data set \( D \) that outputs a total order label.

In the Label Ranking scenario, to measure the degree of corre-
spondence between true and predicted rankings for \( n \)-th example, \( \pi_n \) and \( \hat{\pi}_n \) respectively, it is common to use the Kendall’s tau distance
\( d_{\tau} = \{\{y_n, y_k\} : \pi^{-1}_n(y_j) > \pi^{-1}_n(y_j) \land \pi^{-1}_n(y_j) > \pi^{-1}_n(y_j)\} \). To evaluate a label ranking model, the label ranking loss on the data set
\( D \) is defined as the average normalized Kendall’s tau distance,

\[
loss_{LR} = \frac{1}{N} \sum_{n=1}^{N} \frac{2 \cdot d_{\tau}}{L \cdot (L-1)},
\]  

where \( L \) is the number of labels.

Note that the measure simply counts the number of discordant la-
bel pairs and reports the average over all considered pairwise rank-

ings. Given the general preference matrix representation, assuming
binary matrix predictions \( Y_n \), we can rewrite (1) as

\[
loss_{P} = \frac{1}{N} \sum_{n=1}^{N} \frac{\|Y_n - \hat{Y}_n\|_F^2}{L \cdot (L-1)},
\]  

where \( \| \cdot \|_F \) is the Frobenius matrix norm. Indeed, for each example \( n \), the square of the Frobenius norm sums up to double the number of discordant label pairs.

For models with soft label preference predictions \( \hat{Y}_n \), e.g., \( \hat{Y}_n(i, j) = 0.7, \hat{Y}_n(j, i) = 0.3 \), loss (2) can be interpreted as a
soft version of (1).

We can solve the preference learning task in two stages. In the
learning stage, function \( f : x_n \rightarrow Y_n \), is learned via minimizing
(2). In the aggregation stage, given the model predictions in a form
of \( Y_n \), the total order prediction \( \pi_n \) is computed using a preference
aggregation mapping \( g : Y_n \rightarrow \pi_n \), in the next section we show
the details of the proposed Gaussian Mixture Model algorithm to be
used in the learning stage. Existing algorithms such as [5, 1, 2],
can be used in the aggregation stage.

3 GAUSSIAN MIXTURE MODEL FOR LABEL
RANKING

The GMM model for label ranking is completely defined by a set of
prototypes \((m_k, Q_k)\), for \( k = 1, ..., K \), where \( m_k \) is a d-dimensional vector in input space and \( Q_k \) is the corresponding preference matrix.

First, we introduce the probability \( P(k | x) \) of assigning observa-
tion \( x \) to \( k \)-th prototype that is dependent on their (Euclidean) dis-
tance. Let us assume that the probability density \( P(x) \) can be described by a mixture model,

\[
P(x) = \sum_{k=1}^{K} P(x | k) \cdot P(k),
\]  

where \( K \) is the number of prototypes, \( P(k) \) is the prior probability
that a data point is generated by \( k \)-th prototype, and \( P(x | k) \) is
the conditional probability that \( k \)-th prototype generates particular
data point \( x \). Let us represent the conditional density function
\( P(x | k) \) with the normalized exponential form
\( P(x | k) = \theta(k) \cdot \exp(f(x, m_k)) \) and consider a Gaussian mixture
with \( \theta(k) = (2\pi \sigma^2) \) and \( f(x, m_k) = -||x - m_k||^2/2\sigma^2 \).

We assume that all prototypes have the same standard deviation \( \sigma \) and the same prior
\( P(k) = 1/K \). Given this, using the Bayes’ rule we can write the
assignment probability as

\[
P(k | x) = \sum_{u=1}^{K} \frac{\exp(-||x - m_u||^2/2\sigma^2)}{K \sum_{u=1}^{K} \exp(-||x - m_u||^2/2\sigma^2)}.
\]  

To derive a cost function, we propose the following mixture model
for the posterior probability \( P(Y | x) \),

\[
P(Y | x) = \sum_{k=1}^{K} P(k | x) \cdot P(Y | k).
\]  

Based on this model, example \( x \) is assigned to the prototypes prob-
ionally and its preference matrix is a weighted average of the
prototype preference matrices. The mixture model assumes the con-
ditional independence between \( x \) and \( Y \), given \( k \), \( P(Y | x, k) = P(Y | k) \).

For \( P(k | x) \) we assume the Gaussian distribution from
For the probability of generating a preference matrix \( Y \) by prototype \( k, P(Y \mid k) \), we also assume Gaussian error model with mean \( (Y - Q_k) \) and standard deviation \( \sigma_p \). The resulting cost function \( l(\lambda) \) can be written as the negative log-likelihood,

\[
l(\lambda) = -\frac{1}{N} \sum_{n=1}^{N} \log \sum_{k=1}^{K} p(k \mid x_n) \cdot \mathcal{N}(Y - Q_k, \sigma_p^2),
\]

where \( \lambda = \{ m_k, Q_k, k = 1, ..., P, \sigma_p, \sigma_q \} \) are the model parameters. For the compactness of notation, let us define \( g_{nk} = P(k \mid x_n) \) and \( e_{nk} = \mathcal{N}(Y_n - Q_k, \sigma_q^2) \).

It is important to observe that, after proper normalization, (6) reduces to (2) if examples are assigned to prototypes deterministically. Therefore, it can be interpreted as its soft version. If prototype matrices \( Q_k \) consisted of only 0 and 1 entries (hard label preferences) (6) further reduces to (1).

The objective is to estimate the unknown model parameters, namely prototype positions \( m_k \) and their preference matrices \( Q_k, k = 1, ..., K \). This is done by minimizing the cost function \( l(\lambda) \) with respect to the parameters. This can be achieved in several different ways. If online learning capability is a requirement, one can use the stochastic gradient descent method and obtain the learning rules by calculating derivatives \( \partial l(\lambda) / \partial m_k \) and \( \partial l(\lambda) / \partial Q_k \) for \( k = 1, ..., K \). This results in following rules for \( n \)-th training example,

\[
m_k^{n+1} = m_k^n - \alpha(n)(L_n - e_{nk})g_{nk}(x_n - m_k),
\]

\[
Q_k^{n+1} = Q_k^n - \alpha(n)e_{nk}g_{nk}(x_n - Q_k),
\]

where \( L_n = \sum_{k=1}^{K} g_{nk} \cdot e_{nk} \) and \( \alpha(n) \) is the learning rate.

The resulting model has complexity \( O(NKL) \). Otherwise, the problem can straightforwardly be mapped into Expectation-Maximization (EM) framework following the procedure from [17].

Initialization is done by selecting the first \( P \) training points as the initial prototypes. If any \( Q_k \) prototype preference matrix obtained in such manner contains empty elements, they are replaced with 0.5 entries, as the corresponding labels will initially be treated equally.

GMM model generalizes the training data to produce a representation in terms of prototype vectors and effectively utilizes distances to prototypes as a similarity measure to calculate the predicted label rank. When compared to IB-based algorithms, GMM is a more global model, that aggregates over more data, thus also alleviating influence of noise. Therefore, it is expected to outperform IB-based algorithms, whose performance is highly dependent on the quality of the training data and the presence of outliers, since no abstraction is made during the training phase. We could try to aggregate over more data by considering a large number of neighbors in IB algorithms, however, by doing so we start ignoring distances between the query instance and its neighbors as a similarity measure. This remains to be seen after a proper experimental evaluation.

A disadvantage of the GMM model is that it requires aggregation of the predicted label preference matrix to produce a total order of labels. Luckily, most of the existing algorithms have low complexity, e.g. \( O(L \log L) \) for QuickSort [1]. In our future work we plan to evaluate the pros and cons of different aggregation methods.

4 CONCLUSION AND FUTURE WORK

We introduced an idea of a Gaussian Mixture Model algorithm for Label Ranking. The main advantages of the new method are: (1) it is capable of operating in an online manner, (2) it is memory-efficient since it operates on a predefined budget, (3) it preserves privacy, (4) it could potentially reuse the model when new labels are introduced. There are several avenues which need to be pursued further: (1) experimental evaluation of the proposed method on benchmark data (2) determining the optimal number of prototypes \( K \) using statistical learning theory, (3) low rank approximation of pairwise preference matrices to reduce memory requirements, (4) evaluating different preference matrix aggregation algorithms, (5) applying the algorithm to clustering of label ranking data. In the future work we plan to address these issues.

REFERENCES