On Position Error and Label Ranking Through Iterated Choice

Eyke Hüllermeier
Fakultät für Informatik
Otto-von-Guericke-Universität
Magdeburg

Johannes Fürnkranz
Fachbereich Informatik
Technische Universität
Darmstadt

Jürgen Beringer
Fakultät für Informatik
Otto-von-Guericke-Universität
Magdeburg

Abstract

We consider the problem of learning a *ranking function*, that is a mapping from instances to rankings over a finite number of labels. Our learning method, referred to as *ranking by pairwise comparison* (RPC), first induces pairwise order relations from suitable training data, using a natural extension of so-called pairwise classification. A ranking is then derived from a set of such relations by means of a *ranking procedure*. This paper elaborates on a key advantage of such a decomposition, namely the fact that our learner can be adapted to different loss functions by using different ranking procedures on the same underlying order relations. In particular, the Spearman rank correlation is minimized by using a simple weighted voting procedure. Moreover, we discuss a loss function suitable for settings where candidate labels must be tested successively until a target label is found. In this context, we propose the idea of "empirical conditioning" of class probabilities. A related ranking procedure, called "ranking through iterated choice", is investigated experimentally.

1 Introduction

Prediction problems involving complex outputs and structured output spaces have recently received a great deal of attention within the machine learning literature (e.g., [12]). Problems of that kind are particularly challenging, since the prediction of complex structures such as, say, graphs or trees, is more demanding than the prediction of single values as in classification and regression.

A common problem of this type is *preference learning*, the learning with or from preferences.[1] In the literature, one can identify two different learning scenarios for preference learning [8]: (i) learning from *object preferences*, where the task is to order a set of objects according to training information that specifies the preference relations between a set of training objects (see, e.g., [2]), and (ii) learning from *label preferences*, where the task is to learn a mapping from instances to rankings (total orders) over a finite number of class labels [7]. A corresponding *ranking function* can be seen as an extension of a standard classification function that maps instances to single class labels. In this paper, we focus on the second scenario, but our results can be carried over to the first scenario as well.

In [7], we have introduced a method for learning label preferences that we shall subsequently refer to as *ranking by pairwise comparison* (RPC). This method works in two phases. First, pairwise order relations (preferences) are learned from suitable training data, using a natural extension of so-called *pairwise classification*. Then, a ranking is derived from a set of such orders (preferences) by means of a *ranking procedure*.

The goal of this paper is to show that by using suitable ranking functions, our approach can easily be customized to different performance tasks, that is, to different loss functions for rankings. In fact, the need for a ranking of class labels may arise in different learning scenarios. In this work, we are particularly interested in two types of practically motivated learning problems, one in which the complete ranking is relevant and one in which the predicted ranking serves the purpose of reducing the search effort for finding the single target label.

The remainder of the paper is organized as follows: The problem of preference learning is formally introduced in Section 2, and our pairwise approach is presented in Section 3. In Section 4, the aforementioned types of learning problems are discussed and compared in more detail. The ranking procedures suitable for the two types of problems are then discussed in Sections 5 and 6, respectively. Experimental results are presented in Section 7.

2 Learning from Label Preferences

We consider the following learning problem [8]:

Given:

- a set of *labels* $\mathcal{L} = \{ \lambda_i \mid i = 1 \ldots m \}$
- a set of *examples* $\mathcal{S} = \{ x_k \mid k = 1 \ldots n \}$
- for each training example (instance) x_k:
  - a set of *preferences* $P_k \subseteq \mathcal{L} \times \mathcal{L}$, where $(\lambda_i, \lambda_j) \in P_k$ indicates that label λ_i is preferred over label λ_j for instance x_k.

Find: a function that orders the labels $\lambda \in \mathcal{L}$ for any given example.

We will abbreviate $(\lambda_i, \lambda_j) \in P_k$ with $\lambda_i \succ_{x_k} \lambda_j$ or simply $\lambda_i \succ \lambda_j$ if the particular example x_k doesn't matter or is clear from the context.

The above setting has recently been introduced as *constraint classification* in [9]. As shown in that paper, it is a generalization of several common learning settings, in particular

[1] Space restrictions prevent a thorough review of related work in this paper, but we refer to [6] and recent workshops in this area, e.g., those at NIPS-02, KI-03, SIGIR-03, NIPS-04, and GfKl-05 (the second and fifth organized by the authors).

- *ranking:* Each training example is associated with a total order of the labels.
- *classification:* A single class label λ_x is assigned to each example x; implicitly, this defines the set of preferences $\{\, \lambda_x \succ \lambda \,|\, \lambda \in \mathcal{L} \setminus \{\lambda_x\} \,\}$.
- *multi-label classification:* Each example x is associated with a subset $L_x \subseteq \mathcal{L}$ of labels; implicitly, this defines the preferences $\{\lambda \succ \lambda' \,|\, \lambda \in L_x,\, \lambda' \in \mathcal{L} \setminus L_x\}$.

As mentioned above, we are mostly interested in the first problem, that is in predicting a ranking (complete, transitive, asymmetric relation) of the labels. The ranking $\succ_x$ of an instance x can be expressed in terms of a permutation τ_x of $\{1 \ldots m\}$ such that

$$\lambda_{\tau_x(1)} \succ_x \lambda_{\tau_x(2)} \succ_x \ldots \succ_x \lambda_{\tau_x(m)}. \tag{1}$$

Note that we make the simplifying assumption that all preferences are strict, i.e., we do not consider the case of indifference between labels.

An appealing property of this learning framework is that its input, consisting of *comparative* preference information of the form $\lambda_\imath \succ_x \lambda_\jmath$ (x prefers $\lambda_\imath$ to $\lambda_\jmath$), is often easier to obtain than absolute ratings of single alternatives in terms of *utility degrees*. In this connection, note that knowledge about the complete ranking (1) can be expanded into $m(m-1)/2$ comparative preferences $\lambda_{\tau_x(\imath)} \succ \lambda_{\tau_x(\jmath)}$, $1 \leq \imath < \jmath \leq m$.

3 Learning Pairwise Preferences

The idea of pairwise learning is well-known in the context of classification [5], where it allows one to transform a multi-class classification problem, i.e., a problem involving $m > 2$ classes $\mathcal{L} = \{\lambda_1 \ldots \lambda_m\}$, into a number of *binary* problems. To this end, a separate model (base learner) $\mathcal{M}_{\imath\jmath}$ is trained for each *pair* of labels $(\lambda_\imath, \lambda_\jmath) \in \mathcal{L}$, $1 \leq \imath < \jmath \leq m$; thus, a total number of $m(m-1)/2$ models is needed. $\mathcal{M}_{\imath\jmath}$ is intended to separate the objects with label $\lambda_\imath$ from those having label $\lambda_\jmath$.

At classification time, a query x is submitted to all learners, and each prediction $\mathcal{M}_{\imath\jmath}(x)$ is interpreted as a vote for a label. If classifier $\mathcal{M}_{\imath\jmath}$ predicts $\lambda_\imath$, this is counted as a vote for $\lambda_\imath$. Conversely, the prediction $\lambda_\jmath$ would be considered as a vote for $\lambda_\jmath$. The label with the highest number of votes is then proposed as a prediction.

The above procedure can be extended to the case of preference learning in a natural way [7]. A preference information of the form $\lambda_\imath \succ_x \lambda_\jmath$ is turned into a training example (x, y) for the learner $\mathcal{M}_{ab}$, where $a = \min(\imath, \jmath)$ and $b = \max(\imath, \jmath)$. Moreover, $y = 1$ if $\imath < \jmath$ and $y = 0$ otherwise. Thus, $\mathcal{M}_{ab}$ is intended to learn the mapping that outputs 1 if $\lambda_a \succ_x \lambda_b$ and 0 if $\lambda_b \succ_x \lambda_a$:

$$x \mapsto \begin{cases} 1 & \text{if} \quad \lambda_a \succ_x \lambda_b \\ 0 & \text{if} \quad \lambda_b \succ_x \lambda_a \end{cases} \tag{2}$$

The mapping (2) can be realized by any binary classifier. Alternatively, one might of course employ a classifier that maps into the unit interval $[0, 1]$ instead of $\{0, 1\}$. The output of such a "soft" binary classifier can usually be interpreted as a probability or, more generally, a kind of confidence in the classification. Thus, the closer the output of $\mathcal{M}_{ab}$ to 1, the stronger the preference $\lambda_a \succ_x \lambda_b$ is supported.

A preference learner composed of an ensemble of soft binary classifiers (which can be constructed on the basis of training data in the form of instances with associated partial preferences) assigns a *valued preference relation* $\mathcal{R}_x$ to any (query) instance $x \in \mathcal{X}$:

$$\mathcal{R}_x(\lambda_\imath, \lambda_\jmath) = \begin{cases} \mathcal{M}_{\imath\jmath}(x) & \text{if} \quad \imath < \jmath \\ 1 - \mathcal{M}_{\imath\jmath}(x) & \text{if} \quad \imath > \jmath \end{cases}$$

for all $\lambda_\imath \neq \lambda_\jmath \in \mathcal{L}$.

Given a preference relation $\mathcal{R}_x$ for an instance x, the next question is how to derive an associated ranking τ_x. This question is non-trivial, since a relation $\mathcal{R}_x$ does not always suggest a unique ranking in an unequivocal way. In fact, the problem of inducing a ranking from a (valued) preference relation has received a lot of attention in several research fields, e.g., in fuzzy preference modeling and (multi-attribute) decision making [4]. Besides, in the context of our application, it turned out that the *ranking procedure* used to transform a relation $\mathcal{R}_x$ into a ranking τ_x is closely related to the definition of the quality of a prediction and, hence, to the intended purpose of a ranking. In other words, risk minimization with respect to different loss functions might call for different ranking procedures.

4 Ranking Error versus Position Error

In Section 2, we introduced the problem of predicting a ranking of class labels in a formal way, but did not discuss the semantics of a predicted ranking. In fact, one should realize that such a ranking can serve different purposes. Needless to say, this point is of major importance for the evaluation of a predicted ranking.

In this paper, we are especially interested in two types of practically motivated performance tasks. In the first setting, which is probably the most obvious one, the *complete ranking* is relevant, i.e., the positions assigned to all of the labels. As an example, consider the problem or ordering the questions in a questionnaire. Here, the goal is to maximize a particular respondents' motivation to complete the questionnaire. Another example is learning to predict the best order in which to supply a certain set of stores (route of a truck), depending on external conditions like traffic, weather, purchase order quantities, etc.

In case the complete ranking is relevant, the quality of a prediction should be quantified in terms of a distance measure between the predicted and the true ranking. We shall refer to any deviation of the predicted ranking from the true one as a *ranking error*.

To motivate the second setting, consider a fault detection problem which consists of identifying the cause for the malfunctioning of a technical system. If it turned out that a predicted cause is not correct, an alternative candidate must be tried. A ranking then suggests a simple (trial and error) search process, which successively tests the candidates, one by one, until the correct cause is found [1]. In this scenario, where labels correspond to causes, the existence of a single target label (instead of a target ranking) is assumed. Hence, an obvious measure of the quality of a predicted ranking is the number of futile trials made before that label is found. A deviation of the predicted target label's position from the top-rank will subsequently be called a *position error*.

The main difference between the two types of error is that an evaluation of a full ranking (ranking error) attends to all positions. For example, if the two highest ranks of the true ranking are swapped in the predicted ranking, this is as bad as the swapping of the two lowest ranks.

Note that the position error is closely related to the conventional (classification) error, i.e., the incorrect prediction of the top label. In both cases, we are eventually concerned with predictions for the top rank. In our setting, however, we not only try to maximize the number of correct predictions. Instead, in the case of a misclassification, we also look at the position of the target label. The higher this position, the better the prediction. In other words, we differentiate between "bad" predictions in a more subtle way.

Even though we shall not deepen this point in the current paper, we note that the idea of a position error can of course be generalized to multi-label (classification) problems which assume several instead of a single target label for each instance. There are different options for such a generalization. For example, it makes a great difference whether one is interested in having at least one of the targets on a top rank (e.g., since one solution is enough), or whether all of them should have high positions (resp. none of them should be ranked low). An application of the latter type has recently been studied in [3].

5 Minimizing the Ranking Error

The quality of a model $\mathcal{M}$ (induced by a learning algorithm) is commonly expressed in terms of its *expected loss* or *risk*

$$\mathbb{E}\left(D(y, \mathcal{M}(x))\right), \tag{3}$$

where $D(\cdot)$ is a loss or distance function, $\mathcal{M}(x)$ denotes the prediction made by the learning algorithm for the instance x, and y is the true outcome. The expectation $\mathbb{E}$ is taken over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{Y}$ is the output space (e.g., the set $\mathcal{L}$ of classes in classification).[2]

The simplest loss function, commonly employed in classification, is the 0/1–loss: $D(y, \hat{y}) = 0$ for $y = \hat{y}$ and $= 1$ otherwise. Given this loss function, the optimal (Bayes) prediction for a specific instance x is simply the most probable outcome y. In the classification setting, for example, where $\mathcal{Y} = \mathcal{L}$, this estimate is the class with maximum posterior probability $\mathsf{P}(\lambda_\imath \,|\, x)$.

A straightforward generalization of this principle to the ranking setting, where $\mathcal{Y}$ is the class of rankings over $\mathcal{L}$, leads to the prediction

$$\hat{\tau}_x = \arg\max_{\tau \in \mathcal{S}_m} \mathsf{P}(\tau \,|\, x),$$

where $\mathsf{P}(\tau \,|\, x)$ is the conditional probability of a ranking (permutation) given an instance x, and $\mathcal{S}_m$ denotes the class of all permutations of $\{1 \dots m\}$.

Obviously, the simple 0/1–distance function is a rather crude evaluation measure for rankings, because it assigns the same loss to all rankings that differ from the correct ranking, and does not take into account that different rankings can have different degrees of similarity. For this reason, a number of more sophisticated distance measures for rankings have been proposed in literature.

In general, if $D(\tau, \tau')$ is a measure of the distance between two rankings τ and τ', the risk minimizing prediction is

$$\hat{\tau}_x = \arg\min_{\tau \in S_k} \sum_{\tau \in \mathcal{S}_m} D(\tau, \tau') \cdot \mathsf{P}(\tau' \,|\, x). \tag{4}$$

A frequently used distance measure is the sum of squared rank distances

$$D(\tau', \tau) \stackrel{\text{df}}{=} \sum_{\imath=1}^{m} \left(\, \tau'(\imath) - \tau(\imath) \,\right)^2 \tag{5}$$

which is equivalent to the *Spearman rank correlation*[3]

$$1 - \frac{6D(\tau, \tau')}{m(m^2-1)} \;\in [-1, 1].$$

RPC can yield a risk minimizing prediction for this loss function, if the predictions of the binary classifiers are combined by weighted voting, i.e., the alternatives $\lambda_\imath$ are evaluated by means of the sum of weighted votes

$$S(\lambda_\imath) \;=\; \sum_{\lambda_\jmath \neq \lambda_\imath} \mathcal{R}_x(\lambda_\imath, \lambda_\jmath) \tag{6}$$

and ranked according to these evaluations:

$$\lambda_{\tau_x(1)} \succ_x \lambda_{\tau_x(2)} \succ_x \dots \succ_x \lambda_{\tau_x(m)} \tag{7}$$

with τ_x satisfying $S(\lambda_{\tau_x(\imath)}) \geq S(\lambda_{\tau_x(\imath+1)})$, $\imath = 1 \dots m-1$.[4] This is a particular type of "ranking by scoring" strategy; here, the scoring function is given by (6).

Formally, we can show the following result, which provides a theoretical justification for the voting procedure (6). The proof of this theorem can be found in [11].

Theorem 1 *Using the "ranking by scoring" procedure outlined above, RPC is a risk minimizer with respect to (5) as a loss function. More precisely, with*

$$\mathcal{M}_{\imath\jmath}(x) = \mathsf{P}(\lambda_\imath \succ_x \lambda_\jmath) = \sum_{\tau \,:\, \tau(\jmath) < \tau(\imath)} \mathsf{P}(\tau \,|\, x),$$

the expected distance

$$E(\tau') = \sum_{\tau} p(\tau) \cdot D(\tau', \tau) = \sum_{\tau} p(\tau) \sum_{\imath=1}^{m} (\tau'(\imath) - \tau(\imath))^2$$

becomes minimal by choosing τ' such that $\tau'(\imath) \leq \tau'(\jmath)$ whenever $S(\lambda_\imath) \geq S(\lambda_\jmath)$, where $S(\lambda_\imath)$ is given by (6).

6 Minimizing the Position Error

Despite the fact that (5) is a reasonable loss function for rankings, it is not always appropriate. In particular, it assumes that the *complete* ranking is relevant for the quality of a prediction, which is not the case in connection with the fault detection scenario outlined in Section 4. Here, only the prefix of a ranking τ_x is considered, up to the position of the target label λ_x, while the rest of the prediction is of no importance (since the search procedure stops if λ_x has been found). In this case, the loss function only depends on the rank of λ_x.

More specifically, we define the *position error* as $\tau_x^{-1}(\lambda_x)$, i.e., by the position of the target label λ_x in the ranking τ_x. To compare the quality of rankings of different problems, it is useful to normalize the position error for the number of labels. This *normalized position error* is defined as

$$\frac{\tau_x^{-1}(\lambda_x) - 1}{m - 1} \;\in \{0, 1/(m-1) \dots 1\}, \tag{8}$$

What kind of ranking procedure should be used in order to minimize the risk of a predicted ranking with respect to the position error as a loss function? Intuitively, the candidate labels λ should now be ordered according to their probability $\mathsf{P}(\lambda = \lambda_x)$ of being the target label. Especially, the top-rank (first position) should be given to the label $\lambda_\top$ for which this probability is maximal. Regarding the second rank, recall the fault detection metaphor,

[2] The existence of a probability measure over $\mathcal{X} \times \mathcal{Y}$ must of course be assumed.

[3] This is, of course, a similarity rather than a distance measure.

[4] Ties can be broken arbitrarily.

where the second hypothesis for the cause of the fault is only tested in case the first one turned out to be wrong. In this setting, the second rank should not simply be given to the label with the second highest probability according to the measure $\mathsf{P}_1(\cdot) = \mathsf{P}(\cdot)$. Instead, it must be assigned to the label that maximizes the *conditional* probability $\mathsf{P}_2(\cdot) = \mathsf{P}(\cdot \mid \lambda_x \neq \lambda_\top)$, i.e., the probability of being the target label *given that the first proposal was incorrect.*

At first sight, passing from $\mathsf{P}_1(\cdot)$ to $\mathsf{P}_2(\cdot)$ might appear meaningless from a ranking point of view, since standard probabilistic conditioning (dividing all probabilities by $1 - \mathsf{P}(\lambda_\top)$ and setting $\mathsf{P}(\lambda_\top) = 0$) does not change the order of the remaining labels. One should realize, however, that standard conditioning is not an incontestable updating procedure in our context, simply because $\mathsf{P}_1(\cdot)$ is not a "true" measure over the class labels. Rather, it is only an estimated measure coming from a learning algorithm. Thus, it seems sensible to perform "conditioning" not on the measure itself, but rather on the learner that produced the measure. By this we mean retraining the learner on the original data without the $\lambda_\top$-examples, something that could be paraphrased as *empirical conditioning*. To emphasize that this type of conditioning depends on the data $\mathcal{D}$ and the model assumptions (hypothesis space) $\mathcal{H}$ and, moreover, that it concerns an *estimated* ("hat") probability, the conditional measure $\mathsf{P}_2(\cdot)$ could be written more explicitly as

$$\mathsf{P}_2(\cdot) = \widehat{\mathsf{P}}(\cdot \mid \lambda_x \neq \lambda_\top, \mathcal{D}, \mathcal{M}).$$

To motivate the idea of empirical conditioning, suppose that the estimated probabilities come from a classification tree. Of course, the original tree trained with the complete data will be highly influenced by $\lambda_\top$-examples, and the probabilities assigned by that tree to the alternatives $\lambda \neq \lambda_\top$ might be inaccurate. Retraining a classification tree on a reduced set of data might then lead to more accurate probabilities for the remaining labels, especially since the multi-class problem to be solved has now become simpler (as it involves fewer classes).

A problem of the above "ranking through iterated choice" procedure, that is, the successive selection of alternatives by estimating top-labels from (conditional) probability measures $\mathsf{P}_1(\cdot), \mathsf{P}_2(\cdot) \ldots \mathsf{P}_m(\cdot)$, concerns its computational complexity. In fact, realizing empirical conditioning by retraining a standard multi-class classifier comes down to training such a classifier for (potentially) each subset of the label set $\mathcal{L}$. As will be shown in the following, empirical conditioning can be implemented much more efficiently by our pairwise approach.

6.1 Implementing "ranking through iterated choice" by RPC

What kind of aggregation procedure is suitable for deriving an estimated probability distribution from pairwise classifications resp. valued preference $\mathcal{R}(\lambda_\imath, \lambda_\jmath)$? Let $E_\imath$ denote the event that $\lambda_\imath = \lambda_x$, i.e., that $\lambda_\imath$ is the target label, and let $E_{\imath\jmath} = E_\imath \vee E_\jmath$ (either $\lambda_\imath$ or $\lambda_\jmath$ is the target). Then,

$$(m-1)\,\mathsf{P}(E_\imath) = \sum_{\jmath \neq \imath} \mathsf{P}(E_\imath) = \sum_{\jmath \neq \imath} \mathsf{P}(E_\imath \mid E_{\imath\jmath}) \mathsf{P}(E_{\imath\jmath}), \tag{9}$$

where m is the number of labels. Considering the (pairwise) estimates $\mathcal{R}(\lambda_\imath, \lambda_\jmath)$ as conditional probabilities $\mathsf{P}(E_\imath \mid E_{\imath\jmath})$, we obtain a system of linear equations for the (unconditional) probabilities $\mathsf{P}(E_\imath)$:

$$\begin{aligned} \mathsf{P}(E_\imath) &= \frac{1}{m-1} \sum_{\jmath \neq \imath} \mathcal{R}(\lambda_\imath, \lambda_\jmath) \mathsf{P}(E_{\imath\jmath}) \\ &= \frac{1}{m-1} \sum_{\jmath \neq \imath} \mathcal{R}(\lambda_\imath, \lambda_\jmath) (\mathsf{P}(E_\imath) + \mathsf{P}(E_\jmath)) \end{aligned} \tag{10}$$

In conjunction with the constraint $\sum_{\imath=1}^{m} \mathsf{P}(E_\imath) = 1$, this system has a unique solution provided that $\mathcal{R}(\lambda_\imath, \lambda_j) > 0$ for all $1 \leq \imath, \jmath \leq m$ [13].

Based on this result, the "ranking through iterated choice" procedure suggested above can be realized as follows: First, the system of linear equations (10) is solved and the label $\lambda_\imath$ with maximal probability $\mathsf{P}(E_\imath)$ is chosen as the top-label $\lambda_\top$. This label is then removed, i.e., the corresponding row and column of the relation $\mathcal{R}$ is deleted. To find the second best label, the same procedure is then applied to the reduced relation, i.e., by solving a system of $m-1$ linear equations. This process is iterated until a full ranking has been constructed.

Lemma 1 *In each iteration of the above "ranking through iterated choice" procedure, the correct conditional probabilities are derived.*

Proof: Without loss of generality, assume that λ_m has obtained the highest rank in the first iteration. The information that this label is incorrect, $\lambda_m \neq \lambda_x$, is equivalent to $\mathsf{P}(E_m) = 0$, $\mathsf{P}(E_m \mid E_{\jmath m}) = 0$, and $\mathsf{P}(E_\jmath \mid E_{\jmath m}) = 1$ for all $\jmath \neq m$. Incorporating these probabilities in (10) yields, for all $\imath < m$,

$$\begin{aligned} (m-1)\mathsf{P}(E_\imath) &= \sum_{\jmath=1\ldots m, \jmath \neq \imath} \mathsf{P}(E_\imath \mid E_{\imath\jmath}) \mathsf{P}(E_{\imath\jmath}) \\ &= \sum_{\jmath=1..m-1, \jmath \neq \imath} \mathsf{P}(E_\imath \mid E_{\imath\jmath}) \mathsf{P}(E_{\imath\jmath}) + 1\mathsf{P}(E_{\imath m}) \end{aligned}$$

and as $\mathsf{P}(E_{\imath m}) = \mathsf{P}(E_\imath) + \mathsf{P}(E_m) = \mathsf{P}(E_\imath)$,

$$(m-2)\mathsf{P}(E_\imath) = \sum_{\jmath=1..m-1, \jmath \neq \imath} \mathsf{P}(E_\imath \mid E_{\imath\jmath}) \mathsf{P}(E_{\imath\jmath}).$$

Obviously, the last equation is equivalent to (10) for a system with $m-1$ labels, namely the system obtained by removing the m-th row and column of $\mathcal{R}$. □

As can be seen, the pairwise approach is particularly well-suited for the "ranking through iterated choice" procedure, as it allows for an easy incorporation of the information coming from futile trials. One just has to solve the system of linear equations (10) once more, with some of the pairwise probabilities set to 0 resp. 1 (or, equivalently, solve a smaller system of equations). No retraining of any classifier is required!

Theorem 2 *By ranking the alternative labels according to their (conditional) probabilities of being the top-label, RPC becomes a risk minimizer with respect to the position error (8) as a loss function. That is, the expected loss*

$$E(\tau) = \frac{1}{m-1} \sum_{\imath=1}^{m} (\imath - 1) \cdot \mathsf{P}\left(\lambda_{\tau(\imath)} = \lambda_x\right)$$

becomes minimal for the ranking predicted by RPC.

Proof: This result follows almost by definition. In fact, note that we have

$$E(\tau) \propto \sum_{\imath=1}^{m} \mathsf{P}\left(\lambda_x \notin \{\lambda_{\tau(1)} \ldots \lambda_{\tau(\imath)}\}\right),$$

data	m	PnI	PI	MnI	MI
abalone	28	**3,492**	3,552	4,650	**4,004**
anneal	6	**1,023**	1,024	**1,023**	1,028
audiology	24	**2,668**	3,190	2,310	**2,186**
autos	7	**1,498**	1,502	**1,273**	1,293
balance-scale	3	1,357	**1,294**	1,397	**1,326**
glass	7	1,481	**1,449**	1,547	**1,486**
heart-c	5	1,224	1,224	1,231	1,231
heart-h	5	1,197	1,197	1,197	1,197
hypothyroid	4	**1,006**	1,008	**1,005**	1,007
iris	3	1,073	**1,053**	1,073	**1,053**
lymph	4	1,236	1,236	1,270	**1,250**
primary-tumor	22	**3,516**	3,531	4,254	**3,764**
segment	7	1,045	**1,042**	1,135	**1,042**
soybean	19	**1,183**	1,085	1,205	**1,113**
vehicle	4	1,327	**1,313**	1,411	**1,309**
vowel	11	**1,285**	1,309	2,314	**1,274**
zoo	7	1,178	**1,149**	1,238	**1,099**
letter	26	**1,170**	1,202	2,407	**1,279**

Table 1: Position error for conventional pairwise classification (PnI), iterated pairwise classification (PI), conventional multi-class classificiation (MnI) and iterated multi-class classification (MI) using C4.5 as the base learner.

data	m	PnI	PI	MnI	MI
abalone	28	**3,466**	3,500	4,667	**4,358**
anneal	6	1,020	**1,017**	1,031	**1,028**
audiology	24	**2,863**	3,270	**2,394**	3,274
autos	7	**1,434**	1,449	1,449	**1,376**
balance-scale	3	1,256	1,256	1,406	**1,325**
glass	7	**1,444**	1,463	1,612	**1,486**
heart-c	5	1,218	1,218	1,218	1,218
heart-h	5	1,187	1,187	1,187	1,187
hypothyroid	4	1,007	1,007	1,012	**1,011**
iris	3	1,073	1,073	**1,067**	1,073
lymph	4	**1,291**	1,297	1,284	**1,277**
primary-tumor	22	3,499	**3,472**	4,478	**4,316**
segment	7	**1,059**	1,060	1,131	**1,075**
soybean	19	1,124	**1,073**	1,220	**1,123**
vehicle	4	**1,329**	1,343	1,489	**1,449**
vowel	11	**1,387**	1,423	2,501	**1,516**
zoo	7	1,228	**1,188**	**1,307**	1,327
letter	26	**1,176**	1,188	2,168	**1,375**

Table 2: Position error for conventional pairwise classification (PnI), iterated pairwise classification (PI), conventional multi-class classificiation (MnI) and iterated multi-class classification (MI) using Ripper as the base learner

and that, for each position i, the probability to excess this position when searching for the target λ_x is obviously minimized when ordering the labels according to their (conditional) probabilities. □

7 Empirical Results

Regarding the ranking error, our RPC approach has already been investigated empirically in [7; 10]. In this section, we shall therefore focus on the second type of loss function discussed in the paper, the position error, and present first results for the idea of empirical conditioning and the related "ranking through iterated choice" procedure.

Suppose any multi-class classifier, capable of producing probability estimates for the classes under consideration, to be given as a base learner. Depending on whether or not the multi-class problem is decomposed into a number of pairwise problems (to which the same base learner is of course also applicable), and whether or not the learning procedure is iterated, the following four learning strategies are conceivable.

- **Pairwise, iterated (PI):** This is the "ranking through iterated choice" procedure as outlined in Section 6.1.
- **Pairwise, non-iterated (PnI):** The original problem is decomposed into a number of pairwise problems, but the learning procedure is not iterated, i.e., the probabilities (10) are not recomputed. Instead, these probabilities are only computed *once* and the class labels are ranked according these probabilities.
- **Multi-class, iterated (MI):** The "ranking through iterated choice" procedure is implemented using the base learner in its original (multi-class) version.
- **Multi-class, non-iterated (MnI):** A ranking is produced by applying the base learner to the complete data set and ordering the class labels according to their probabilities.

As an aside, let us note that, in connection with selecting the top-label or ordering the labels according to their probability, ties are always broken through coin flipping.

As mentioned before, the strategy MI is tremendously inefficient from a computational point of view, since $m-1$ multi-class classifiers have to be trained (and applied to the query case) in order to produce a single ranking of m labels: The first classifier is trained on the complete data, the second on the reduced data which does not include the examples of the first top-label, and so forth. Although this approach is hardly practical for real applications, we include it in our experiments as our main interest is to compare iterated with non-iterated learning. Our main goal is to find out whether the idea of iterating the learning procedure is beneficial or not.

Tables 1 and 2 show the results that we obtained for a number of well-known benchmark data sets from the UCI repository[5], and the StatLib archive[6], using C4.5 and Ripper as base learners, respectively. For each data set and each method we estimate the mean (absolute) position error using leave-one-out cross validation, except for the data set 'letter', for which we used the predefined separation into training and test data. Table 3 shows the numbers of wins and losses for each pair of methods.

	PnI	PI	MnI	MI
PnI	—	9/6	13/3	9/8
PI	6/9	—	13/4	8/7
MnI	3/13	4/13	—	3/13
MI	9/9	7/8	13/3	—
PnI	—	9/4	13/3	12/3
PI	4/9	—	12/3	13/2
MnI	3/13	3/12	—	3/13
MI	3/12	2/13	13/3	—

Table 3: Win/Loss statistics for each pair of methods, using C4.5 (top) and Ripper (bottom) as base learners.

[5] http://www.ics.uci.edu/~mlearn

[6] http://stat.cmu.edu/

In contrast to our expectations, the results suggest that iterating (empirical conditioning) does not pay off in the pairwise learning approach. More often than not, the average position error for the non-iterated variant is smaller than the one for the iterated version. On the other hand, empirical conditioning significantly outperforms standard conditioning in the case of multi-class classification (the results are significant at a level of 2% according to a simple sign test).

Even though the results do not comply with our first expectations, they can be explained in an intuitively plausible way. In fact, one has to realize that the idea of empirical conditioning produces two antagonistic effects:

- In each iteration, the size of the data set to learn from becomes smaller. This reduction of data comes along with a loss of information.
- Due to the reduced number of classes, the learning problems become simpler in each iteration.

The first effect will have a negative influence on generalization performance, whereas the second one will have a positive influence. In pairwise classification, the first effect manifests itself by a reduction of the number of "voters": In the k-th iteration of iterated choice, only $m-k+1$ labels and, hence, only $(m-k+1)(m-k)/2$ binary classifiers participate. Since the score of each label is thus derived from the votes of only $m-k$ instead of $m-1$ such classifiers, the impact of each individual binary classifier on the final ranking increases. An erroneous prediction of a single binary classifier will have a larger effect if fewer classifiers are used to derive the ranking. Thus, the ranking scores become less reliable with a decreasing number of labels.

For conventional iterated choice, this is countered by the fact that the classifiers become increasingly simple, because it can be expected that the decision boundary for separating m classes is more complex than the decision boundary for separating $m' < m$ classes of the same problem. The crucial point is that this effect is effectively disabled in the pairwise approach: Since the original learning problem is decomposed into pairwise problems right from the start, the simplification due to a reduction of class labels is already bailed out at the beginning. In later iterations, some pairwise problems become irrelevant, but the remaining problems do not become simpler. Consequently, only the first (negative) effect remains, which in turn explains the deterioration for the pairwise approach. In contrast, the second (positive) effect often seems to dominate the first effect in the case of multi-class classification.

8 Concluding Remarks

By showing that RPC is a risk minimizer with respect to particular loss functions for rankings, this paper provides a sound theoretical foundation for ranking by pairwise comparison. The interesting point is that RPC can easily be customized to different performance tasks, simply by changing the ranking procedure employed in the second step of the method. By modifying this procedure, the goal of RPC can be changed from minimizing the expected distance between the predicted and the true ranking to minimizing the expected number of futile trials in searching a target label. This can be done without retraining the classifier ensemble.

The second type of loss function, the position error, is minimized by ordering the class labels according their (conditional) probability of being the target label. To improve the estimations of these probabilities, we proposed the idea of "empirical conditioning" and the related "ranking through iterated choice" procedure. In an experimental study, this procedure was compared with the standard ("non-iterated") variant where the probabilities are not recomputed, i.e., where the class labels are ranked according to the originally estimated probabilities. Our results suggest that empirical conditioning does indeed reduce the expected loss in the case of standard multi-class classification (where the "choice" of the top-label is realized by a multi-class classifier in each iteration), whereas it does not pay off in the case of pairwise learning. Even though this finding is interesting by itself and gives some important theoretical insights, it is to some extent unfortunate from a practical point of view, since empirical conditioning can be implemented efficiently only in the pairwise approach.

References

[1] C. Alonso, J.J. Rodríguez, and B. Pulido. Enhancing consistency based diagnosis with machine learning techniques. In *Current Topics in AI*, pp. 312–321. Springer, 2004.

[2] W.W. Cohen, R.E. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.

[3] K. Crammer and Y. Singer. A family of additive online algorithms for category ranking. *Journal of Machine Learning Research*, 3:1025–1058, 2003.

[4] J. Fodor, M. Roubens. *Fuzzy Preference Modelling and Multicriteria Decision Support*. Kluwer, 1994.

[5] J. Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2:721–747, 2002.

[6] J. Fürnkranz and E. Hüllermeier. *Pairwise preference learning and ranking—Proceedings of the KI-2003 Workshop*, Technical Report TR-2003-14, Österreichisches Forschungsinstitut für Artificial Intelligence, Wien, 2003.

[7] J. Fürnkranz and E. Hüllermeier. Pairwise preference learning and ranking. In *Proc. ECML-03*, Cavtat-Dubrovnik, Croatia, 2003.

[8] J. Fürnkranz and E. Hüllermeier. Preference learning. *Künstliche Intelligenz*, 1/05:60–61, 2005.

[9] S. Har-Peled, D. Roth, and D. Zimak. Constraint classification: a new approach to multiclass classification. In *Proc. ALT-02*, pp. 365–379, Lübeck, 2002.

[10] E. Hüllermeier and J. Fürnkranz. Comparison of ranking procedures in pairwise preference learning. In *Proc. IPMU-04*, Perugia, 2004.

[11] E. Hüllermeier and J. Fürnkranz. Learning label preferences: Ranking error versus position error. In *Proc. IDA–2005*, Madrid, 2005.

[12] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proc. ICML–2004*, pp. 823–830, Banff, Alberta, 2004.

[13] T.F. Wu, C.J. Lin, and R.C. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5:975–1005, 2004.