Learning Preference Models from Data: On the Problem of Label Ranking and Its Variants*

Eyke Hüllermeier† and Johannes Fürnkranz‡

† FB Mathematik und Informatik, Philipps-Universität Marburg, Germany
‡ FB Informatik, TU Darmstadt, Germany

Abstract The term "preference learning" refers to the application of machine learning methods for inducing preference models from empirical data. In the recent literature, corresponding problems appear in various guises. After a brief overview of the field, this work focuses on a particular learning scenario called label ranking, where the problem is to learn a mapping from instances to rankings over a finite number of labels. Our approach for learning such a ranking function, called ranking by pairwise comparison (RPC), first induces a binary preference relation from suitable training data, using a natural extension of pairwise classification. A ranking is then derived from this relation by means of a ranking procedure. This paper elaborates on a key advantage of such an approach, 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

Recently, the topic of *preferences* has attracted considerable attention in Artificial Intelligence (AI) research, notably in fields such as agents, nonmonotonic reasoning, constraint satisfaction, planning, and qualitative decision theory Doyle (2004). Preferences provide a means for specifying desires in a declarative way, which is a point of critical importance for AI.

*Extended version of the conference paper Hüllermeier and Fürnkranz (2005).

In fact, consider AI's paradigm of a rationally acting (decision-theoretic) agent: The behavior of such an agent has to be driven by an underlying preference model, and an agent recommending decisions or acting on behalf of a user should clearly reflect that user's preferences. Therefore, the formal modeling of preferences can be considered an essential aspect of autonomous agent design.

Drawing on past research on knowledge representation and reasoning, AI offers qualitative and symbolic methods for treating preferences that can reasonably complement traditional approaches that have been developed for quite a while in fields such as economic decision theory. Needless to say, however, the acquisition of preferences is not always an easy task. Therefore, not only are modeling languages and representation formalisms needed, but also methods for the automatic learning, discovery and adaptation of preferences. For example, computerized methods for discovering the preferences of individuals are useful in e-commerce and various other fields where an increasing trend toward personalization of products and services can be recognized.

It is hence hardly surprising that methods for learning and predicting preferences in an automatic way are among the very recent research topics in disciplines such as machine learning, knowledge discovery, and recommender systems. In these fields, several approaches have been subsumed under the terms of ranking and preference learning. In Section 2, we distinguish between *object ranking problems*, where the task is to order a set of objects according to a preference function, and *label ranking problems*, where the task is to assign a permutation of a fixed set of class labels to a given object. We observe that there are two principal approaches to address these preference learning tasks. One possibility is to try to learn a *utility function* which in turn induces the sought ranking. The underlying assumption is that the observed preferences are based on a hidden numerical preference model which produces a numerical score for each object or label. The second approach is to model a *binary preference relation* that, instead of evaluating *single* objects, allows for comparing *pairs* of alternatives. The assumption here is that a total order of all alternatives can be derived from pairwise preferences between these alternatives. In a learning context, this approach is complicated by the fact that preferences learned from data are not necessarily correct and, hence, may be inconsistent (e.g., violate transitivity).

Among the four problem types that can be obtained by applying the two modeling approaches to the two learning tasks, this paper will focus of the fourth option, which appears to be the most recent one considered in the literature. More specifically, the learning scenario that we will consider in

this paper consists of a collection of training examples which are associated with a finite set of decision alternatives. Following the common notation of supervised learning, we shall refer to the latter as *labels*. However, contrary to standard classification, a training example is not assigned a single label, but a set of *pairwise preferences* between labels (which neither has to be complete nor entirely consistent), each one expressing that one label is preferred to another. The goal is to use pairwise preferences from training examples for predicting a total order, a *ranking*, of all possible labels for a new training example. The *ranking by pairwise comparison* (RPC) algorithm, to be introduced and investigated in this paper, works in two phases. First, pairwise 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 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: We start with an overview of preference learning and a systematic presentation of ranking problems in Section 2. Subsequently, our pairwise approach to label ranking will be presented in Section 3. In Section 4, the aforementioned variants of the label ranking problem 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. The paper ends with some concluding remarks in Section 8.

2 Preference Learning

In this section, we will motivate preference learning as a theoretically interesting and practically relevant subfield of machine learning. Moreover, we will give a systematic overview of existing approaches in this field.

When talking about preference learning, two questions immediately arise: First, whose preferences are to be learned? And second, in which way are the preferences formalized, i.e., what kind of model is to be inferred?

Regarding the first question, one can basically distinguish between learning the preferences of a *single* individual and learning those of a *group* of individuals (or, more generally, agents or instances). Both types of prob-

lems are relevant. When attempting to personalize a web search engine, for example, one is preliminary interested in the preferences of a particular user. On the other hand, in collaborative filtering one tries to predict the preferences of any client on the basis of the preferences of other clients. Note that this presupposes a connection between individuals which allows one to model dependencies. In fact, learning the preferences of a group of individuals only makes sense if these individuals' preferences are not completely independent of each other. (Otherwise, one actually faces multiple instances of the first type of problem, namely learning the preferences of each of the individuals separately.) In collaborative filtering, such a connection is established on the basis of a client's *preference profile*, which allows one to distinguish similar clients from dissimilar ones.

In the following, we shall more formally distinguish two types of learning problems, namely *learning from object preferences* and *learning from label preferences*. The corresponding scenarios are roughly in agreement with the above distinction between learning the preferences of a single individual and those of a group of individuals: When learning object preferences, only a single preference relation is considered. Here, the focus is on the objects, which are characterized through attributes. When learning label preferences, one tries to induce many preference relations simultaneously. Here, the individuals are represented through attributes, while the objects are considered as unrelated items. A more formal definition of these learning scenarios will be given in Sections 2.1 and 2.2 below.

Regarding the second question, there are two approaches for dealing with preferences that prevail the literature on choice and decision theory. These are based, respectively, on the two perhaps most natural ways for expressing preferences, namely by *evaluating* individual alternatives, or by *comparing* (pairs of) competing alternatives. In the latter approach, binary relations are employed in order to express (comparative) preferences in a qualitative way. The basic relation, often denoted $\succeq$, is interpreted as a weak preference, i.e., $a \succeq b$ means that "alternative a is at least as good as alternative b".

The second approach is more inclined to numerical representations of preferences via *utility* or *value functions*. A utility function $f(\cdot)$ assigns an abstract utility degree to each alternative under consideration and thus induces a preference relation $\succeq$ by virtue of $a \succeq b \Leftrightarrow f(a) \geq f(b)$. Obviously, the numerical approach provides stronger information than the relational one but is also more restrictive and more demanding from a modeling point of view. Advantages and disadvantages of these two approaches, from a learning point of view, will be discussed in Sections 2.3 and 2.4.

The two answers to the above two questions, respectively, give rise to

	modeling utility functions	modeling pairwise preferences
object ranking	comparison training Tesauro (1989)	learning to order things Cohen et al. (1998)
label ranking	constraint classification Har-Peled et al. (2002)	**this work** Fürnkranz and Hüllermeier (2003)

Table 1. Four different approaches to learning from preference information together with representative references.

the four combinations shown in Table 1. Despite their differences in the underlying models, all these approaches assume the same type of training information. This information is of the relational type, that is, a single piece of information specifies that one entity (object or label) is preferred to some other entity.

Before discussing these four settings in more detail, let us note that the term "preference" should not be taken literally and instead always be interpreted in a wide sense. Thus, $a \succ b$ can indeed mean that alternative a is more liked by a person than b, but also, for example, that a is an algorithm that outperforms b on a certain problem, that a is an event which is more probable than b, or that a is a student finishing her studies before b.

2.1 Learning from Object Preferences

The most frequently studied problem in learning from preferences is to induce a *ranking function* $r(\cdot)$ that is able to order any subset $\mathcal{O}$ of an underlying class $\mathcal{X}$ of objects. That is, $r(\cdot)$ assumes as input a subset $\mathcal{O} = \{x_1 \dots x_n\} \subseteq \mathcal{X}$ of objects and returns as output a permutation τ of $\{1 \dots n\}$. The interpretation of this permutation is that object x_i is preferred to x_j whenever $\tau(i) < \tau(j)$. The objects themselves are typically characterized by a finite set of features as in conventional attribute-value learning. The training data consists of a set of exemplary pairwise preferences. This scenario is also known as "learning to order things" Cohen et al. (1998).

As an example consider the problem of learning to rank query results of a search engine Joachims (2002); Radlinski and Joachims (2005). The training information is provided implicitly by the user who clicks on some

of the links in the query result and not on others. This information can be turned into binary preferences by assuming that the selected pages are preferred over nearby pages that are not clicked on Joachims et al. (2005).

2.2 Learning from Label Preferences

In this learning scenario, the problem is to predict, for any instance x (e.g., a person) from an instance space $\mathcal{X}$, a preference relation $\succ_x \subseteq \mathcal{L} \times \mathcal{L}$ among a finite set $\mathcal{L} = \{\lambda_1 \dots \lambda_m\}$ of labels or alternatives, where $\lambda_i \succ_x \lambda_j$ means that, for the input x, label λ_i is preferred to label λ_j. More specifically, as we are especially interested in the case where $\succ_x$ is a total strict order,[1] the problem is to predict a permutation of $\mathcal{L}$.

The training information consists of a set of instances for which (partial) knowledge about the associated preference relation is available. More precisely, each training instance x is assumed to be associated with only a *subset* of all pairwise preferences which, moreover, can be conflicting. Therefore, this setting covers the typical situation in practice where only incomplete preference information is known. In particular, a ranking model can be learned from standard *classification* data in which each instance is associated with only a single label λ^*: Considering this label as the maximally preferred one gives rise to the set of $(m-1)$ preferences $\{\lambda^* \succ \lambda \,|\, \lambda \in \mathcal{L}\}$.

2.3 Learning Utility Functions

As mentioned above, one natural way to represent preferences is to evaluate individual alternatives by means of a (real-valued) utility function. In the object preferences scenario, such a function is a mapping $f : \mathcal{X} \to \mathbb{R}$ that assigns a utility degree $f(x)$ to each object x and, hence, induces a total order on $\mathcal{X}$. In the label preferences scenario, a utility function $f_i : \mathcal{X} \to \mathbb{R}$ is needed for each of the labels λ_i, $i = 1 \dots m$. Here, $f_i(x)$ is the utility assigned to alternative λ_i by instance x. To obtain a ranking for x, the alternatives are ordered according to these utility scores, i.e., $\lambda_i \succeq_x \lambda_j \Leftrightarrow f_i(x) \geq f_j(x)$.

If the training data would offer the utility scores directly, preference learning would reduce to a regression problem (up to a monotonic transformation of the utility values). This information can rarely be assumed, however. Instead, only constraints derived from comparative preference information of the form "This object (or label) should have a higher utility score than that object (or label)" are usually given. Thus, the challenge for

[1] Needless to say, these are strong assumption that will often not be satisfied in practice; we shall come back to this issue in the concluding remarks in Section 8.

the learner is to find a function (from an underlying hypothesis space) that is as much as possible in agreement with all constraints.

For object ranking approaches, this idea has first been formalized in Tesauro (1989) under the name *comparison training*. Here, the author proposed a symmetric neural-network architecture that can be trained with representations of two states and a training signal that indicates which of the two states is preferable. The elegance of this approach comes from the property that one can replace the two symmetric components of the network with a single network, which can subsequently provide a real-valued evaluation of single states. Similarly, training data in the form of pairwise comparisons of objects is used in Haddawy et al. (2003) to train a neural network. The network learns a function that takes two objects as input and outputs either 0 or 1, depending on whether or not the first object is preferred to the second one. The structure of the network is not arbitrary, but is chosen in a way that allows one to combine domain knowledge about partial preferences between the feature-based state descriptions of the objects. This work is based on an earlier approach proposed in Wang (1994).

Similar ideas have also been investigated for training other types of classifiers, in particular support vector machines. We already mentioned Joachims (2002), where "click-through data" was analyzed in order to rank documents retrieved by a search engine according to their relevance. Earlier, an algorithm for training SVMs from pairwise preference relations between objects was proposed in Herbrich et al. (1998).

For the case of label ranking, a corresponding method for learning the functions $f_i(\cdot)$, $i = 1 \dots m$, from training data has been proposed in the framework of *constraint classification*, introduced as an extension of standard classification in Har-Peled et al. (2002, 2003). The learning method proposed in this work constructs two training examples for each preference $\lambda_i \succ \lambda_j$, where the original N-dimensional training examples are mapped into a $(m \times N)$-dimensional space. The positive example copies the original training vector into the components $((i-1)N+1) \dots iN$ and its negation into the components $((j-1)N+1) \dots jN$ of a vector in the new space. The remaining entries are filled with 0, and the negative example has the same elements with reversed signs. In this $(m \times N)$-dimensional space, the learner tries to find a separating hyperplane. For classifying a new example x_0, the labels are ordered according to the response resulting from multiplying x_0 with the i-th N-element section of the hyperplane vector.

2.4 Learning Preference Relations

It has already been noted that *observed* preferences are usually of the relational type, since utility scores are difficult to elicit. For example, it is very hard to ensure a consistent scale even if all utility evaluations are performed by the same user, let alone if utility scores are elicited from different users which may not have a uniform scale of their scores Cohen et al. (1998). In Pyle (1999), the author claims that it is easier for a human to determine an order between several items if one makes pairwise comparisons between the individual items and then adds up the wins for each item, instead of trying to order the items right away.

For the *learning* of preferences, one may bring up a similar argument. It will typically be easier to learn a separate theory for each individual preference that compares two objects or two labels and determines which one is better. This technique essentially reduces the problem of learning preferences to a binary classification task. The learner is trained to predict for each pair of objects/labls which one is preferable. In Pyle (1999), a very similar technique called *pairwise ranking* is proposed in order to facilitate human decision-making in ranking problems. Of course, every learned utility function that assigns a score to a set of labels $\mathcal{L}$ induces such a binary preference relation on these labels.

Note that the preference relation induced by a utility function is necessarily a total order, whereas the converse is not true: not every learned binary preference relation induces a total order that can be represented by a utility function. The point is that the learned preference relation does not necessarily have the typical properties of order relations, for example, it is not necessarily transitive. In fact, to obtain a ranking (total strict order) of the items, one has to apply a *ranking procedure* that takes a binary preference relation as input and produces a ranking as output. The simplest approach is *ranking by scoring*: A score, such as the number of pairwise comparisons in which an item is preferred, is derived from the binary relation, and the items are then ordered according to their scores. For alternative (more complex) combination schemes see e.g. Wu et al. (2004); Hüllermeier and Fürnkranz (2004).

For object ranking problems, the pairwise approach has been pursued in Cohen et al. (1998). This paper proposes to solve object ranking problems by learning a binary preference predicate $Q(x, x')$, which predicts whether x is preferred to x' or vice versa. A final ordering is found in a second phase by deriving a ranking that is maximally consistent with these predictions (in the sense of violating as few as possible binary preferences).

For label ranking problems, the pairwise approach has been introduced by the authors in Fürnkranz and Hüllermeier (2003). The key idea, that we

shall describe in more detail below, is to learn, for each pair of labels (λ_i, λ_j), a binary predicate $\mathcal{M}_{ij}(x)$ that predicts whether $\lambda_i \succ_x \lambda_j$ or $\lambda_j \succ_x \lambda_i$ for an input x. In order to rank the labels for a new object, predictions for all pairwise label preferences are obtained and a ranking that is maximally consistent with these preferences is derived. This approach is a natural extension of pairwise classification, i.e., the idea to tackle a multi-class classification problem by learning separate theories for each pair of classes.

3 Ranking by Pairwise Comparison

As mentioned above, the idea of pairwise learning is well-known in the context of classification Fürnkranz (2002), where it allows one to transform a multi-class classification problem, i.e., a problem involving $m > 2$ classes $\mathcal{L} = \{\lambda_1 \dots \lambda_m\}$, into a number of *binary* problems. To this end, a separate model (base learner) $\mathcal{M}_{ij}$ is trained for each *pair* of labels $(\lambda_i, \lambda_j) \in \mathcal{L}$, $1 \leq i < j \leq m$; thus, a total number of $m(m-1)/2$ models is needed. $\mathcal{M}_{ij}$ is intended to separate the objects with label λ_i from those having label λ_j.

At classification time, a query x is submitted to all learners, and each prediction $\mathcal{M}_{ij}(x)$ is interpreted as a vote for a label. If classifier $\mathcal{M}_{ij}$ predicts λ_i, this is counted as a vote for λ_i. Conversely, the prediction λ_j would be considered as a vote for λ_j. 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 Fürnkranz and Hüllermeier (2003). A preference information of the form $\lambda_i \succ_x \lambda_j$ is turned into a training example (x, y) for the learner $\mathcal{M}_{ab}$, where $a = \min(i, j)$ and $b = \max(i, j)$. Moreover, $y = 1$ if $i < j$ 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 $\lambda_b \succ_x \lambda_a$:

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

The mapping (1) 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 $x \in \mathcal{X}$:

$$\mathcal{R}_x(\lambda_i, \lambda_j) = \begin{cases} \mathcal{M}_{ij}(x) & \text{if} \quad i < j \\ 1 - \mathcal{M}_{ij}(x) & \text{if} \quad i > j \end{cases} \tag{2}$$

for all $\lambda_i \neq \lambda_j \in \mathcal{L}$. Given a preference relation of that kind, 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 Fodor and Roubens (1994). 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.

The approach introduced in this section, referred to as *ranking by pairwise comparison* (RPC), consists of two steps and can be summarized as follows:

- First, a valued preference relation (2) is derived by training an ensemble of (soft) binary classifiers, and then
- a ranking of the labels $\mathcal{L}$ is obtained by means of a suitable ranking procedure.

4 Ranking Error versus Position Error

In Section 3, we introduced the problem of predicting a ranking of class labels in a formal way, but without discussing the semantics of a predicted ranking. In fact, one should realize that 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 of 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 Alonso et al. (2004). 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 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.

5 Minimizing the Ranking Error

The quality of a model $\mathcal{M}$, induced by a learning algorithm from a set of data $\mathcal{D}$, is commonly expressed in terms of its *expected loss* or *risk*

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

where $\mathcal{M}(x)$ 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] $D(\cdot)$ is a (real-valued) loss or distance function $\mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$.

In the context of label ranking, $\mathcal{Y}$ can formally be defined by the class $\mathcal{S}_m$ of all permutations of $\{1 \dots m\}$. A frequently used distance measure on

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

this class is the sum of squared rank distances:

$$D(\tau', \tau) \stackrel{\text{df}}{=} \sum_{i=1}^{m} \big(\tau'(i) - \tau(i) \big)^2 \tag{4}$$

for all $\tau, \tau' \in \mathcal{S}_m$. The measure (4) is closely related to the well-known *Spearman rank correlation*, a similarity measure which is obtained by the linear transformation

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

Given a distance measure $D(\cdot)$ for rankings, the risk minimizing prediction for an instance x is

$$\hat{\tau}_x = \arg\min_{\tau \in S_k} \sum_{\tau \in \mathcal{S}_m} D(\tau, \tau') \cdot \mathbb{P}(\tau' \,|\, x), \tag{5}$$

where $\mathbb{P}(\tau \,|\, x)$ is the conditional probability of a ranking (permutation) given x.

RPC can yield a risk minimizing prediction for the loss function (4). To this end, the predictions of the binary classifiers have to be combined by weighted voting, i.e., the alternatives λ_i are evaluated by means of the sum of weighted votes

$$S(\lambda_i) = \sum_{\lambda_j \neq \lambda_i} \mathcal{R}_x(\lambda_i, \lambda_j) \tag{6}$$

and ranked according to these evaluations:

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

with τ_x satisfying $S(\lambda_{\tau_x(i)}) \geq S(\lambda_{\tau_x(i+1)})$, $i = 1 \ldots m-1$.[3] This is a particular type of "ranking by scoring" strategy, with the scoring function 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 Hüllermeier and Fürnkranz (2005).

Theorem 5.1. *Using the "ranking by scoring" procedure outlined above, RPC is a risk minimizer with respect to (4) as a loss function. More precisely, given that*

$$\mathcal{M}_{ij}(x) = \mathbb{P}(\lambda_i \succ_x \lambda_j) = \sum_{\tau \,:\, \tau(j) < \tau(i)} \mathbb{P}(\tau \,|\, x),$$

[3] Ties can be broken arbitrarily.

the expected distance ($p(\cdot)$ is a probability distribution on $\mathcal{S}_m$)

$$E(\tau') = \sum_{\tau \in \mathcal{S}_m} p(\tau) \cdot D(\tau', \tau) = \sum_{\tau \in \mathcal{S}_m} p(\tau) \sum_{i=1}^{m} (\tau'(i) - \tau(i))^2$$

becomes minimal by choosing τ' such that $\tau'(i) \leq \tau'(j)$ whenever $S(\lambda_i) \geq S(\lambda_j)$, where $S(\lambda_i)$ is given by (6).

6 Minimizing the Position Error

Despite the fact that (4) 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 for problems involving different numbers of labels, it is useful to employ a *normalized position error* which is defined as

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

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 $\mathbb{P}(\lambda = \lambda_x)$ 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, 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 $\mathbb{P}_1(\cdot) = \mathbb{P}(\cdot)$. Instead, it must be assigned to the label that maximizes the *conditional* probability $\mathbb{P}_2(\cdot) = \mathbb{P}(\cdot \mid \lambda_x \neq \lambda_\top)$, i.e., the probability of being the target label *given that the first proposal was incorrect.*

At first sight, passing from $\mathbb{P}_1(\cdot)$ to $\mathbb{P}_2(\cdot)$ might appear meaningless from a ranking point of view, since standard probabilistic conditioning (dividing

all probabilities by $1 - \mathbb{P}(\lambda_\top)$ and setting $\mathbb{P}(\lambda_\top) = 0$) 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 $\mathbb{P}_1(\cdot)$ 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 $\mathbb{P}_2(\cdot)$ could be written more explicitly as

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

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 $\mathbb{P}_1(\cdot), \mathbb{P}_2(\cdot) \ldots \mathbb{P}_m(\cdot)$, 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 preferences $\mathcal{R}(\lambda_i, \lambda_j)$? Let E_i denote the event that $\lambda_i = \lambda_x$, i.e., that λ_i is the target label, and let $E_{ij} = E_i \vee E_j$ (either λ_i or λ_j is the target). Then,

$$(m-1)\,\mathbb{P}(E_i) = \sum_{j \neq i} \mathbb{P}(E_i) = \sum_{j \neq i} \mathbb{P}(E_i \,|\, E_{ij}) \mathbb{P}(E_{ij}), \tag{9}$$

where m is the number of labels. Considering the (pairwise) estimates $\mathcal{R}(\lambda_i, \lambda_j)$ as conditional probabilities $\mathbb{P}(E_i \mid E_{ij})$, we obtain a system of linear equations for the (unconditional) probabilities $\mathbb{P}(E_i)$:

$$\begin{aligned}\mathbb{P}(E_i) &= \frac{1}{m-1} \sum_{j \neq i} \mathcal{R}(\lambda_i, \lambda_j) \mathbb{P}(E_{ij}) \\ &= \frac{1}{m-1} \sum_{j \neq i} \mathcal{R}(\lambda_i, \lambda_j) (\mathbb{P}(E_i) + \mathbb{P}(E_j)) \end{aligned} \tag{10}$$

In conjunction with the constraint $\sum_{i=1}^{m} \mathbb{P}(E_i) = 1$, this system has a unique solution provided that $\mathcal{R}(\lambda_i, \lambda_j) > 0$ for all $1 \leq i, j \leq m$ Wu et al. (2004).

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 λ_i with maximal probability $\mathbb{P}(E_i)$ 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 6.1. *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 $\mathbb{P}(E_m) = 0$, $\mathbb{P}(E_m \mid E_{jm}) = 0$, and $\mathbb{P}(E_j \mid E_{jm}) = 1$ for all $j \neq m$. Incorporating these probabilities in (10) yields, for all $i < m$,

$$\begin{aligned}(m-1)\mathbb{P}(E_i) &= \sum_{j=1\ldots m, j \neq i} \mathbb{P}(E_i \mid E_{ij}) \mathbb{P}(E_{ij}) \\ &= \sum_{j=1..m-1, j \neq i} \mathbb{P}(E_i \mid E_{ij}) \mathbb{P}(E_{ij}) + 1\mathbb{P}(E_{im})\end{aligned}$$

and as $\mathbb{P}(E_{im}) = \mathbb{P}(E_i) + \mathbb{P}(E_m) = \mathbb{P}(E_i)$,

$$(m-2)\mathbb{P}(E_i) = \sum_{j=1..m-1, j \neq i} \mathbb{P}(E_i \mid E_{ij}) \mathbb{P}(E_{ij}).$$

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 6.2. *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*

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

becomes minimal for the ranking predicted by RPC.

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

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

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 Fürnkranz and Hüllermeier (2003); Hüllermeier and Fürnkranz (2004). 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.

Table 2. Win/Loss statistics for each pair of methods, using C4.5 (top) and Ripper (bottom) as base learners.

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

- **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.

Table 3 show the results that we obtained for a number of well-known

Table 3. 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 (left) and Ripper (right) as the base learner (better performance indicated in bold).

C4.5					
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**

Ripper					
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**

benchmark data sets from the UCI repository and the StatLib archive[4], 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 2 shows the numbers of wins and losses for each pair of methods.

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 conventional conditioning in the case of multi-class classification (the results are significant at a level of 2% according to a simple sign test which tests the hypothesis that the approaches have equal expected performance on all data sets).

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:

- Information loss: 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.
- Simplification: 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 RPC, 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 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 prob-

[4] http://www.ics.uci.edu/ mlearn, http://stat.cmu.edu/

lems 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.

Despite the fact that the idea of empirical conditioning (iterative learning) does apparently not pay off in the case of pairwise learning, one should note that, according to Table 2, the best pairwise method, namely the non-iterated version, is nevertheless better than the best multi-class method, namely the iterated one.

8 Concluding Remarks

In this paper, we have first given an overview of recent research activities in a subfield of machine learning that we have called *preference learning*. Roughly speaking, preference learning is concerned with learning preference models from observed data. More specifically, we have focused on an especially simple class of preference models, namely *rankings*. In particular, we have studied the problem of *label ranking* in more detail.

To approach the label ranking problem, we have proposed an extension of pairwise classification, called ranking by pairwise comparison (RPC). By showing that RPC is a risk minimizer with respect to quadratic 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. As an explanation, we offered the observation that the simplification effect (classifying becomes simpler if fewer classes are involved) does not grab in the pairwise approach or, stated differently, this effect is already fully exploited from the outset. And indeed, the non-iterated version of the pairwise approach is still superior to iterated multi-class classification, while being computationally much more efficient.

Regarding future work, there are many avenues and open problems. For example, instead of looking at particular types of loss functions as we have done in this paper, it would be interesting to find a more complete characterization of the type of loss functions for rankings that can can be minimized by the pairwise learning approach and those that cannot. Another open issue is an extension of label ranking to the learning of more general preference relations on the label set $\mathcal{L}$. In fact, in many practical applications it might be reasonable to relax the assumption of strictness, i.e., to allow for indifference between labels, or even to represent preferences in terms of partial instead of total orders.

Acknowledgments: This research was supported by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG).

Bibliography

C. Alonso, JJ. Rodríguez, and B. Pulido. Enhancing consistency based diagnosis with machine learning techniques. In *Proc. 10th Conference of the Spanish Association for Artificial Intelligence*, pages 312–321. Springer, 2004.

William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*. The MIT Press, 1998.

Jon Doyle. Prospects for preferences. In *Computational Intelligence*, volume 20, pages 111–136, 2004.

J. Fodor and M. Roubens. *Fuzzy Preference Modelling and Multicriteria Decision Support*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994.

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

J. Fürnkranz and E. Hüllermeier. Pairwise preference learning and ranking. In *Proc. ECML–2003, 13th European Conference on Machine Learning*, Cavtat-Dubrovnik, Croatia, September 2003.

P. Haddawy, V. Ha, A. Restificar, B. Geisler, and J. Miyamoto. Preference elicitation via theory refinement. *Journal of Machine Learning Research*, 4:317–337, 2003.

S. Har-Peled, D. Roth, and D. Zimak. Constraint classification: a new approach to multiclass classification. In *Proceedings 13th Int. Conf. on Algorithmic Learning Theory*, pages 365–379, Lübeck, Germany, 2002. Springer.

Sariel Har-Peled, Dan Roth, and Dav Zimak. Constraint classification for multiclass classification and ranking. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15 (NIPS-02)*, pages 785–792, 2003.

R. Herbrich, T. Graepel, P. Bollmann-Sdorra, and K. Obermayer. Supervised learning of preference relations. In *Proceedings FGML-98, German National Workshop on Machine Learning*, pages 43–47, 1998.

E. Hüllermeier and J. Fürnkranz. Comparison of ranking procedures in pairwise preference learning. In Proc. IPMU–04, Perugia, Italy, 2004.

Eyke Hüllermeier and Johannes Fürnkranz. Learning label preferences: Ranking error versus position error. In *Advances in Intelligent Data Analysis VI*, Madrid, 2005. Springer.

T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-02)*, pages 133–142. ACM Press, 2002.

T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR-05)*, 2005.

Dorian Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann, San Francisco, CA, 1999.

F. Radlinski and T. Joachims. Learning to rank from implicit feedback. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD-05)*, 2005.

Gerald Tesauro. Connectionist learning of expert preferences by comparison training. In D. Touretzky, editor, *Advances in Neural Information Processing Systems 1 (NIPS-88)*, pages 99–106. Morgan Kaufmann, 1989.

J. Wang. Artificial neural networks versus natural neural networks: a connectionist paradigm for preference assessment. *Decision Support Systems*, 11:415–429, 1994.

TF. Wu, CJ. Lin, and RC. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5:975–1005, 2004.