Preference Learning and Ranking by Pairwise Comparison

Johannes Fürnkranz[1] and Eyke Hüllermeier[2]

[1] Technische Universität Darmstadt, Germany
juffi@ke.tu-darmstadt.de
[2] Philipps-Universität Marburg, Germany
eyke@mathematik.uni-marburg.de

Abstract. This chapter provides an overview of recent work on preference learning and ranking via pairwise classification. The *learning by pairwise comparison* (LPC) paradigm is the natural machine learning counterpart to the relational approach to preference modeling and decision making. From a machine learning point of view, LPC is especially appealing as it decomposes a possibly complex prediction problem into a certain number of learning problems of the simplest type, namely binary classification. We explain how to approach different preference learning problems, such as label and instance ranking, within the framework of LPC. We primarily focus on methodological aspects, but also address theoretical questions as well as algorithmic and complexity issues.

1 Introduction

Preferences on a set of alternatives can be expressed in two natural ways, namely by evaluating *individual* and by comparing *pairs* of alternatives. As for human decision making, the comparative (relational) approach is intuitively appealing and, moreover, supported by evidence from cognitive psychology. Indeed, instead of directly choosing one alternative from a set of options, or ranking all alternatives directly according to their desirability, it is often much simpler to start by comparing alternatives in a pairwise fashion. The actual problem, whether choosing a single best alternative or ranking all of them, is then solved in a second step on the basis of these pairwise comparisons. Essentially, this is known as Thurstone's *Law of Comparative Judgment* [43]. Modern decision-theoretic methodologies, such as the *Analytic Hierarchy Process* [38], are often based on pairwise comparisons between different options in various stages of complex decision processes [39].

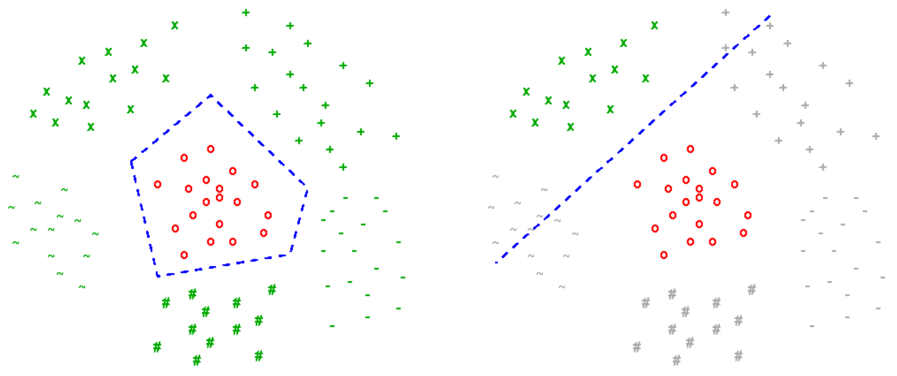
The decomposition of the original problem into a set of presumably simpler subproblems is not only advantageous for human decision making but also useful from a machine learning point of view. In fact, as will be argued in more detail later on, the resulting learning problems can typically be solved in a more accurate and efficient way. The price to pay is a possibly more involved prediction step. Roughly speaking, the pairwise comparisons, being made independently of each other, can be conflicting, so that their aggregation into a solution of the original problem may become non-trivial. On the other hand, such a two-step procedure —learning a binary comparison relation and aggregating the pairwise comparisons into a (complex) prediction afterward— also has another advantage: It is *modular* in the sense that the learning part is decoupled from the prediction part; thus, it becomes possible to solve different types of prediction problems on the basis of the same learning algorithm (ensemble of binary classifiers), simply by changing the aggregation procedure in the second step.

The idea of *learning by pairwise comparison* (LPC), or simply *pairwise learning*, has been explored quite extensively for conventional classification, where it is known under a variety of names such as *all pairs*, *1-vs-1*, or *round robin* learning. Here, it is used as a special binarization technique, that is, to decompose a polychotomous classification problem into a set of pairwise problems, thereby making multi-class problems amenable to binary classification methods. Motivated by its successful use for classification as well as its intuitive appeal from a preference and decision making perspective, the LPC approach has been extended to different types of preference learning and ranking problems in recent years. The purpose of this chapter is to give an overview of existing work and recent developments in this line of research.

In Section 2, we briefly recall the use of pairwise learning in conventional classification. This approach can be generalized in a quite natural way to the setting of label ranking, as will be explained in Section 3. The use of LPC for label ranking has in turn motivated its application for a number of generalized classification problems, and these will be discussed in Section 4. Section 5 is devoted to the application of pairwise learning for the problem of instance ranking. Section 6 reviews some formal results regarding the optimality of LPC for specific prediction problems, and Section 7 addresses the aspect of complexity. Finally, we conclude this chapter with a short summary and an outlook on future work in Section 8.

2 LPC for Classification

The use of the pairwise approach to preference learning, the main topic of this chapter, is motivated by its successful application in conventional classification. As a special type of binary decomposition technique, it allows one to tackle multi-class problems with binary classifiers. The key idea is to transform a k-class problem involving classes $\mathcal{Y} = \{y_1, y_2, \ldots, y_k\}$ into $k(k-1)/2$ binary problems, one for each pair of classes. More specifically, a separate model (base learner) $\mathcal{M}_{i,j}$ is trained for each pair of labels $(y_i, y_j) \in \mathcal{Y} \times \mathcal{Y}$, $1 \le i < j \le k$, using the examples from these two classes as their training set; thus, a total number of $k(k-1)/2$ models is needed. $\mathcal{M}_{i,j}$ is intended to separate the objects with label y_i from those having label y_j. At classification time, a query instance $\boldsymbol{x} \in \mathcal{X}$ is submitted to all models $\mathcal{M}_{i,j}$, and their predictions $\mathcal{M}_{i,j}(\boldsymbol{x})$ are combined into an overall prediction. In the simplest case, each prediction $\mathcal{M}_{i,j}(\boldsymbol{x})$ is interpreted as a vote for either y_i or y_j, and the label with the highest number of votes is proposed as a final prediction.



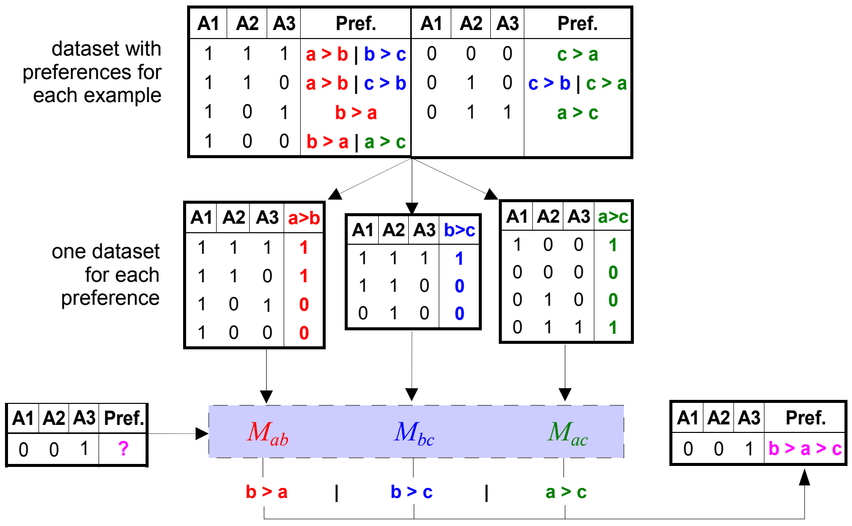
(a) *One-against-all class binarization* (transforms each k-class problem into k binary problems, one for each class, where each of these problems uses the examples of its class as the positive examples (here o), and all other examples as negatives.

(b) *Pairwise class binarization* transforms each k-class problem into $k(k-1)/2$ binary problems, one for each pair of classes (here 0 and x) ignoring the examples of all other classes.

Fig. 1. Class Binarization Methods

In comparison to alternative decomposition techniques, such as the one-vs-all approach which learns one model for each label, the pairwise decomposition facilitates effective learning as it leads to maximally simple problems. In particular, the pairwise problems are computationally less complex, since each of them contains fewer training examples (because all examples that do not belong to either of the two classes are ignored). Perhaps even more importantly, these problems typically have simpler decision boundaries. This is illustrated in the example shown in Figure 1, where each pair of classes can be separated with a linear decision boundary, while more complex functions are required to separate each class from all other classes. Evidence supporting the conjecture that the decision boundaries of the binary problems are indeed simpler can also be found in practical applications: In [26], it was observed that the classes of a digit recognition task were pairwise linearly separable, while the corresponding one-against-all task was not amenable to single-layer networks. Similarly, in [16] the authors obtained a larger advantage of pairwise classification over one-vs-all for support vector machines with a linear kernel than for support vector machines with a non-linear kernel.

The basic idea of pairwise classification is fairly well-known from the literature. It has been used in the areas of statistics [3, 8], neural networks [25, 26, 37, 31], support vector machines [40, 15, 28, 16], and others. We refer to [9] for a brief survey of the literature on this topic.


dataset with preferences for each example
one dataset for each preference
M_{ab}
M_{bc}
M_{ac}
b > a | b > c | a > c

A1	A2	A3	Pref.
1	1	1	a > b | b > c
1	1	0	a > b | c > b
1	0	1	b > a
1	0	0	b > a | a > c
0	0	0	c > a
0	1	0	c > b | c > a
0	1	1	a > c

A1	A2	A3	a>b
1	1	1	1
1	1	0	1
1	0	1	0
1	0	0	0

A1	A2	A3	b>c
1	1	1	1
1	1	0	0
0	1	0	0

A1	A2	A3	a>c
1	0	0	1
0	0	0	0
0	1	0	0
0	1	1	1

A1	A2	A3	Pref.
0	0	1	?

A1	A2	A3	Pref.
0	0	1	b > a > c

Fig. 2. Schematic illustration of learning by pairwise comparison.

3 LPC for Label Ranking

In this learning scenario, the problem is to predict, for any instance $\boldsymbol{x}$ (e.g., a person) from an instance space $\mathcal{X}$, a ranking of a finite set $\mathcal{Y} = \{y_1, y_2, \ldots, y_k\}$ of labels or alternatives (e.g. politicians in an election), i.e., a total order relation $\succ_{\boldsymbol{x}} \subseteq \mathcal{Y} \times \mathcal{Y}$ where $y_i \succ_{\boldsymbol{x}} y_j$ means that instance $\boldsymbol{x}$ prefers the label y_i to the label y_j.

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 $\boldsymbol{x}$ is associated with a subset of all pairwise preferences $y_i \succ_{\boldsymbol{x}} y_j$, $1 \leq i, j \leq k$. The top of Figure 2 shows a training set consisting of seven examples, each described in terms of three attributes $A1$, $A2$, and $A3$. Each training example is associated with some preferences over the set of possible labels $\mathcal{Y} = \{a, b, c\}$. For example, for the second training instance, we know that $a \succ b$ and $c \succ b$, but we do not know whether a or c is the most preferred option. Thus, even though we assume the existence of an underlying ("true") ranking, we do not expect the training data to provide full information about that ranking. Besides, in order to increase the practical usefulness of the approach, we even allow for inconsistencies, such as pairwise preferences which are conflicting (cyclic) due to observation errors.

3.1 Learning Preference Relations

Pairwise classification (cf. Section 2) can be extended to the above problem of learning from label preferences in a natural way [11]. To this end, a preference (order) information of the form $y_r \succ_{\boldsymbol{x}} y_s$ is turned into a training example $(\boldsymbol{x}, z)$ for the learner $\mathcal{M}_{ij}$, where $i = \min(r, s)$ and $j = \max(r, s)$. Moreover, $z = 1$ if $r < s$ and $z = 0$ otherwise. Thus, $\mathcal{M}_{ij}$ is intended to learn the mapping that outputs 1 if $y_i \succ_{\boldsymbol{x}} y_j$ and 0 if $y_j \succ_{\boldsymbol{x}} y_i$:

$$\boldsymbol{x} \mapsto \begin{cases} 1 & \text{if} \quad y_i \succ_{\boldsymbol{x}} y_j \\ 0 & \text{if} \quad y_j \succ_{\boldsymbol{x}} y_i \end{cases} . \tag{1}$$

The model is trained with all examples $\boldsymbol{x}$ for which either $y_i \succ_{\boldsymbol{x}} y_j$ or $y_j \succ_{\boldsymbol{x}} y_i$ is known. Examples for which nothing is known about the preference between y_i and y_j are ignored.

The mapping (1) can be realized by any binary classifier. Alternatively, one may also employ base classifiers that map into the unit interval $[0, 1]$ instead of $\{0, 1\}$, and thereby assign a *valued preference relation* $\mathcal{R}_{\boldsymbol{x}}$ to every (query) instance $\boldsymbol{x} \in \mathcal{X}$:

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

for all $y_i \neq y_j \in \mathcal{Y}$. The output of a $[0, 1]$-valued classifier can usually be interpreted as a probability or, more generally, a kind of confidence in the classification: the closer the output of $\mathcal{M}_{ij}$ to 1, the stronger the preference $y_i \succ_{\boldsymbol{x}} y_j$ is supported.

3.2 An Example

Figure 2 illustrates the entire process. First, the original training set is transformed into three two-class training sets, one for each possible pair of labels, containing only those training examples for which the relation between these two labels is known. Then three binary models, $\mathcal{M}_{ab}$, $\mathcal{M}_{bc}$, and $\mathcal{M}_{ac}$ are trained. In our example, the result could be simple rules like the following:

$$\begin{aligned} &\mathcal{M}_{ab} : a > b \ \ \textbf{if } A2 = 1. \\ &\mathcal{M}_{bc} : b > c \ \ \textbf{if } A3 = 1. \\ &\mathcal{M}_{ac} : a > c \ \ \textbf{if } A1 = 1 \vee A3 = 1. \end{aligned}$$

Given a new example with an unknown preference structure (shown in the bottom left of Figure 2), the predictions of these models are then used to predict a ranking for this example. As we will see in the next section, this is not always as trivial as in this example.

3.3 Combining Predicted Preferences into a Ranking

Given a predicted preference relation $\mathcal{R}_{\boldsymbol{x}}$ for an instance $\boldsymbol{x}$, the next question is how to derive an associated ranking. This question is non-trivial, since a relation $\mathcal{R}_{\boldsymbol{x}}$ does not always suggest a unique ranking in an unequivocal way. For example, the learned preference relation is not necessarily transitive. 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 [6]. In the context of pairwise classification and preference learning, several studies have empirically compared different ways of combining the predictions of individual classifiers [44, 1, 20, 10].

A simple though effective strategy is a generalization of the aforementioned voting strategy: each alternative y_i is evaluated by the sum of (weighted) votes

$$S(y_i) = \sum_{j \neq i} \mathcal{R}_{\boldsymbol{x}}(y_i, y_j), \tag{3}$$

and all labels are then ordered according to these evaluations, i.e., such that

$$(y_i \succ_{\boldsymbol{x}} y_j) \Rightarrow (S(y_i) \geq S(y_j)). \tag{4}$$

Even though this ranking procedure may appear rather ad-hoc at first sight, it does have a theoretical justification (cf. Section 6).

4 LPC for Generalized Classification Problems

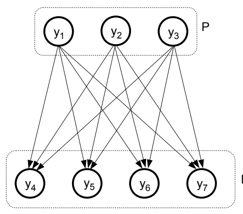
It has been observed by several authors [14, 11, 5] that, in addition to classification, many learning problems, such as multilabel classification, ordered classification, or ranking may be formulated in terms of label preferences. In this section, we summarize work on using pairwise learning to address such problems.

4.1 Multilabel Classification

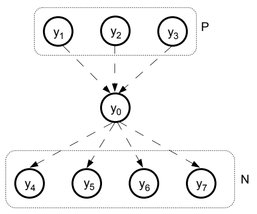
Multilabel classification refers to the task of learning a function that maps instances $\boldsymbol{x} \in \mathcal{X}$ to label subsets $\mathcal{P}_{\boldsymbol{x}} \subset \mathcal{Y}$, where $\mathcal{Y} = \{y_1, y_2, \ldots, y_k\}$ is a finite set of predefined labels, typically with a small to moderate number of alternatives. Thus, in contrast to multiclass learning, alternatives are not assumed to be mutually exclusive such that multiple labels may be associated with a single instance. The set of labels $\mathcal{P}_{\boldsymbol{x}}$ are called *relevant* for the given instance, the set $\mathcal{N}_{\boldsymbol{x}} = \mathcal{Y} \setminus \mathcal{P}_{\boldsymbol{x}}$ are the *irrelevant* labels.

In conventional label ranking, a training example typically consists of an instance $\boldsymbol{x} \in \mathcal{X}$, represented in terms of a fixed set of features, and a set of pairwise preferences over labels $\mathcal{R}_{\boldsymbol{x}} \subset \mathcal{Y} \times \mathcal{Y}$, where $(y, y') \in \mathcal{R}_{\boldsymbol{x}}$ is interpreted as $y \succ_{\boldsymbol{x}} y'$. In multilabel classification, the training information consists of a set $\mathcal{P}_{\boldsymbol{x}}$ of relevant labels and, implicitly, a set $\mathcal{N}_{\boldsymbol{x}} = \mathcal{Y} \setminus \mathcal{P}_{\boldsymbol{x}}$ of irrelevant labels. The idea of applying methods for (pairwise) label ranking in the context of multilabel classification is based on the observation that this information can be expressed equivalently in terms of a set of preferences (cf. Figure 3 (a)):

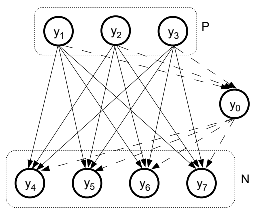
$$\hat{\mathcal{R}}_{\boldsymbol{x}} = \{(y, y') \mid y \in \mathcal{P}_{\boldsymbol{x}} \wedge y' \in \mathcal{N}_{\boldsymbol{x}}\} \tag{5}$$



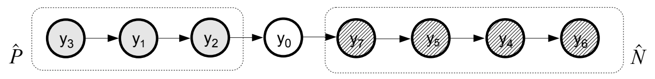
(a) the set of preferences representing a multilabel classification problem



(b) introducing a calibration label y_0 that separates $\mathcal{P}$ and $\mathcal{N}$



(c) the set of preferences representing a calibrated label ranking problem



(d) at prediction time, the calibration label y_0 indicates the split into labels that are predicted relevant ($\hat{\mathcal{P}}$) and labels that are predicted irrelevant ($\hat{\mathcal{N}}$)

Fig. 3. Calibrated Label Ranking

In fact, this representation is in a sense even more flexible than the original one; for example, it easily remains applicable in the case where the relevance of some labels is unknown. The preferences (5) can be used to train a pairwise label ranker, which is then able to predict a ranking over all possible labels of a new, unseen example.

Note, however, that a ranking, while determining an order of the labels, does actually not define a partitioning into subsets of relevant and irrelevant labels. A natural way to obtain such a partitioning as additional information is to find an appropriate split-point t in the ranking, suggesting that the first t labels in the ranking are relevant while the remaining ones are irrelevant. A "calibrated" ranking of that kind nicely combines two types of prediction, namely a label ranking and a multilabel classification [4]. From a ranking point of view, it covers additional information about *absolute* preferences; from the point of view of multilabel classification, it offers additional order information, i.e., information about the *relative* preferences within the two sets of relevant and irrelevant labels.

In [45], a few straight-forward approaches for determining such split-points are discussed, including methods for determining a fixed threshold for all examples or individual thresholds for each possible label. However, none of these approaches allows one to adjust the thresholds for each individual example.

In [4, 12], we therefore proposed to incorporate the split-point into the learning process. This was achieved by introducing an artificial (neutral) label which is associated with the split-point and thus calibrates the ranking: For each training instance, this label is preferred over all irrelevant labels, but is less preferable than all relevant labels; see Figure 3 (b). A *calibrated label ranker* trained on the enhanced set of preferences (Figure 3 (c)) is then able to predict a ranking of all labels, *including the artificial one.* The position of this label indicates where the ranking has to be split into a relevant and an irrelevant part. Experiments have shown that this approach outperforms standard methods for multilabel classification, despite a slight tendency to underestimate the number of relevant labels.

4.2 Ordered and Hierarchical Classification

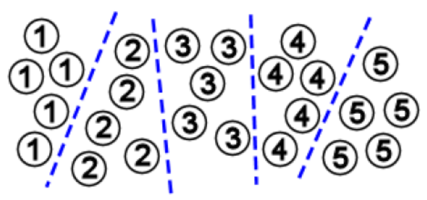
Ordered classification and *hierarchical classification* are problems in which the target label set has an inherent structure. In ordered classification, this structure is a total order, such as $small < medium < large$. In hierarchical problems, the structure is a partial order in the form of a hierarchy, typically defined by various subconcept/superconcept relations; for example, $Hessen < Germany < Europe$ and $Bayern < Germany < Europe$ (while *Hessen* and *Bayern* are incomparable, i.e., neither $Hessen < Bayern$ nor $Bayern < Hessen$).

The use of conventional loss functions, such as the 0/1 loss, is obviously questionable in the context of ordered and hierarchical classification, as it does not take the relation between class labels into account. If *small* is the true class, for instance, then *medium* is a better prediction than *large*, despite the fact that both are incorrect. An obvious idea, then, is to express this type of information in the form of *preferences* on labels.

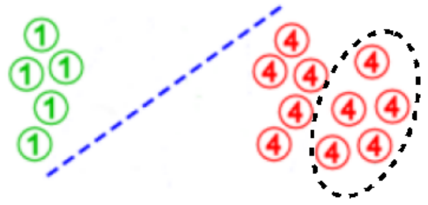
Within the framework of LPC, this can be done by associating a training instance with all pairwise preferences that can be *inferred* from the label structure. For example, if we know that an instance is of class *small*, we not only know that the label *small* is preferred over all other labels, but we can also infer that *medium* would be a better classification than *large*. In other words, the set of training examples for which it is known that $medium \succ large$ could be *enriched* by instances from the class *small*. Similarly, if we know that an object belongs to the class *Hessen*, we can infer that the label *Rheinland-Pfalz* would be preferred over *Lower Austria*, because the former is also a German state, while the latter is in a different country.

One idea is to directly add such inferred preferences to the pairwise training data [33]. Figure 4 illustrates this for a problem with five classes, which are labeled from 1 to 5. Here, the classifier $\mathcal{M}_{1,4}$ is enriched with examples from class 5, for which the prediction 4 is clearly preferred to the prediction 1. Note that the examples of class 3 are *not* added to class 4: Since the underlying scale is only ordinal but not numerical, it is not legitimate to assume that 3 is "closer" to 4 than to 1 and, therefore, that predicting 4 for an example from class 3 is preferred to predicting 1.

An analogous technique can be used for hierarchical classification, where the preferences can defined via common ancestors in the hierarchy. Interestingly, in [41] it was shown that this method may be viewed as a generalization of the



(a) an ordered classification problem where the order of the classes is reflected in the instance space



(b) the training set for the classifier $\mathcal{M}_{1,4}$ is enriched with the examples of class 5.

Fig. 4. Enriched pairwise classification in ordered domains (figures taken from [33])

so-called Pachinko-machine classifier [27], which contains one binary classifier for each internal node of the label hierarchy.

Even though the above idea of using pairwise preference learning for ordinal and hierarchical classification has not yet been explored in full depth, first experimental results suggest that it does not improve the performance of the conventional pairwise classifier. There is a number of possible reasons and explanations for this. Notably, enriching the training data comes with a loss of a key advantage of the pairwise approach, namely the simplicity of the binary problems and the decision boundaries of the binary classifiers. The more inferred preferences are added, the more complex the decision boundary of a single binary classifier will become, eventually approaching the complex decision boundaries of a one-against-all classifier. Moreover, unlike the example in Figure 4(a), it seems that in many hierarchical and ordered classification problems, the structure on the labels is not so clearly reflected in the topology of the instance space. This means that examples that are "close" in label space are not necessarily neighbored in instance space.

This aspect has been investigated in more detail in [17]. This work also aimed at answering the question to what extent existing techniques and learning algorithms for ordered classification are able to exploit order information, and which properties of these techniques are important in this regard. The main conclusions that could be drawn from this study are as follows: Most learning techniques for ordinal classification are indeed able to exploit order information about classes if such information is present, which is often the case though not always. An important factor in this regard is the flexibility of the learner. Roughly speaking, the less flexible a learner, the more it benefits from an ordinal structure. Interestingly enough, it was found that pairwise classification is fully competitive to other meta-learning techniques specifically designed for ordered classification problems [7]. This result is surprising, since pairwise classification, in its original form, does not explicitly exploit an ordinal structure (and, compared to the other techniques, even uses a smaller amount of training information in terms of the total number of training examples). However, by training only on pairs of classes, it is trivially consistent with each ordinal structure. In a sense, one can argue that it exploits ordinal structure in an implicit way whenever this is possible, but as its binary problems are not explicitly tailored toward the assumption of an ordinal structure, it does not deteriorate when this assumption is invalid.

5 LPC for Instance Ranking

5.1 Multipartite Ranking

Recall that the term *instance ranking* is used in this book as a generic term of bipartite and multipartite ranking. Multipartite ranking proceeds from the setting of ordinal classification, where an instance $\boldsymbol{x} \in \mathcal{X}$ belongs to one among a finite set of classes $\mathcal{Y} = \{y_1, y_2, \ldots, y_k\}$ and, moreover, the classes have a natural order: $y_1 < y_2 < \ldots < y_k$. Training data consists of a set $\mathcal{T}$ of labeled instances.

In contrast to the classification setting, however, the goal is not to learn a classifier but a ranking function $f(\cdot)$. Given a subset $X \subset \mathcal{X}$ of instances as an input, the function produces a ranking of these instances as an output (typically by assigning a score to each instance and then sorting by scores). A prediction of this type is evaluated in terms of ranking measures such as the C-index. Ideally, a ranking is produced in which instances from higher classes precede those from lower classes.

5.2 LPC for Multipartite Ranking

The use of LPC for multipartite ranking was recently proposed in [13]. As usual, a separate model $\mathcal{M}_{i,j}$ is induced for each pair of classes $(y_i, y_j) \in \mathcal{Y} \times \mathcal{Y}$, $1 \leq i < j \leq k$, using the subset $\mathcal{T}_{i,j} \subset \mathcal{T}$ of examples from these classes as training data. At classification time, a query $\boldsymbol{x}$ is submitted to all models.

To predict a ranking of a set of query instances $X \subset \mathcal{X}$, each instance $\boldsymbol{x} \in X$ is scored in terms of an aggregation of the predictions

$$\{ \mathcal{M}_{i,j}(\boldsymbol{x}) \,|\, 1 \leq i, j \leq k \} \ .$$

More specifically, the score is defined as the (weighted) sum of the predictions "in favor of a higher class", that is

$$f(\boldsymbol{x}) \;=\; \sum_{1 \leq i < j \leq k} (p_i + p_j)\, f_{j,i}(\boldsymbol{x}) = \sum_{1 \leq i < j \leq k} (p_i + p_j)\, \mathcal{M}_{j,i}(\boldsymbol{x}) \ , \tag{6}$$

where p_i is the probability of class y_i (estimated by the relative frequency in the training data).

In first experimental studies, this approach has been compared to state-of-the-art ranking methods such as SVMRank [24] which, instead of decomposing the original problem into a set of small binary classification problems, transforms it into a single, large classification problem. The results suggest that LPC is competitive in terms of predictive accuracy while being much more efficient from a computational point of view. Roughly speaking, the reason is that solving several small problems is typically more efficient than solving a single large one (cf. Section 7). In this particular case, the transformation into a single classification problem is especially critical, as it may come with a considerable increase of the number of original training examples.

6 Theoretical Foundations

Despite their intuitive appeal and practical success, it is of course important to justify pairwise learning methods from a theoretical point of view. In particular, one may wonder whether LPC is provably able to produce predictions that are optimal in the sense of minimizing (the expectation of) a given loss function.

Corresponding results have recently been derived, not only for classification (Section 6.1) but also for ranking (Sections 6.2 and 6.3). Apart from further technical assumptions, which are not detailed here, these results typically rely on the idealized assumption that the prediction of a pairwise model, $\mathcal{M}_{i,j}(\boldsymbol{x})$, can be interpreted as a *probability*, for example the probability that, in the label ranking associated with $\boldsymbol{x}$, label y_i precedes label y_j. Needless to say, assumptions of that kind are not always easy to satisfy in practice, especially because probability estimation is a challenging problem potentially more difficult than classification.

Besides, the pairwise approach suffers from an inherent limitation, which is sometimes called the "non-competence" problem in the literature. Roughly speaking, this problem is caused by the fact that a pairwise model is only trained on parts of the instance space and, therefore, possibly not competent for inputs coming from other parts. In conventional classification, for example, a pairwise model $\mathcal{M}_{i,j}$ is only trained on examples from classes y_i and y_j and, therefore, arguably non-competent for classifying instances from other classes. Several proposals for addressing this problem can be found in the literature. In [32] it was proposed to combine the pairwise models $\mathcal{M}_{i,j}$ with a separate set of models $\mathcal{M}_{ij,\bar{\mathcal{Y}}_{ij}}$, which predict whether an example belongs to the classes y_i or y_j, or to the remaining classes $\bar{\mathcal{Y}}_{ij} = \mathcal{Y} \setminus \{y_i, y_j\}$. A similar proposal is to learn ternary models $\mathcal{M}_{i,j,\bar{\mathcal{Y}}_{ij}}$, which directly discriminate between the three options y_i, y_j, or none of the two [2]. Both approaches have to sacrifice one of the key advantages of the pairwise approach, namely the simplicity of the learned binary models.

6.1 Classification

Despite the existence of more sophisticated methods, such as *pairwise coupling* [15, 44], the most popular strategy for aggregating the predictions of pairwise classifiers is "voting". In binary voting, each classifier $\mathcal{M}_{i,j}$ can give a "vote" for either y_i or y_j. In weighted voting, it may split its vote among the two classes, for example according to its probability estimate. Having queried all models, the class with the highest number of votes (sum of weighted votes) is eventually predicted.

Empirically, weighted voting is known to perform very well. A theoretical justification for this performance has been derived in [23]. Under some technical assumptions, it was shown there that weighted voting may be considered as an approximation of a generalized voting strategy, called *adaptive voting*, which in turn was shown to be optimal in the sense of yielding a maximum posterior probability (MAP) prediction of the true class. Besides, weighted voting appears to be even more robust than adaptive voting in the sense of being less sensitive toward violations of the underlying model assumptions.

Moreover, it was shown that the pairwise approach to learning, at least in a slightly generalized form, is able to produce Bayes-optimal decisions in the context of conventional classification [42].

6.2 Label Ranking

In the context of label ranking, it was shown that many standard loss functions on rankings can be minimized in expectation [22]. For example, under some technical assumptions, straight-forward weighted voting is a risk minimizer for the sum of squared rank distances as a loss function (thus, it maximizes the expected Spearman rank correlation between the true and the predicted label ranking). Replacing weighted voting by another aggregation strategy, one can also minimize the number of pairwise inversions (i.e., maximize Kendall's tau); the aggregation problem itself, however, is NP-hard.

On the other hand, there are also loss functions that cannot be minimized by LPC. Roughly speaking, this is due to a loss of information caused by decomposing the original problem into a set of pairwise problems. Examples include the Spearman footrule and Ulam's distance [21].

6.3 Position Error

It may also be reasonable to compare a predicted ranking with a single target label instead of target ranking. For example, given a predicted label ranking, the *position error* is defined by the position on which the true class label is found or, stated differently, the number of labels that are ranked ahead of this target label. In a normalized form, this measure directly generalizes the conventional 0/1-loss for classification, assuming a value of 0 (1) if the target label is put on the first (last) position. The problem of predicting a ranking that minimizes the expected position loss can again be solved in a theoretically optimal way by LPC [21].

Practically, it was shown that the problem can be reduced to an iterated classification problem, which in turn can be solved effectively through a procedure called *empirical conditioning*. This procedure amounts to iteratively predicting a label and re-training the classifier on the remaining labels. While this is practically infeasible in the general case (essentially one needs to train one classifier for each label subset), it can be realized efficiently by means of LPC. This is because, in the pairwise approach, re-training of the classifiers is not necessary; instead, only the aggregation phase needs to be changed [21].

7 Complexity

At first sight, it seems that LPC is very inefficient for high numbers of labels because one has to train a quadratic number of classifiers. However, a closer look reveals that this is often outweighed by a positive effect, namely that the individual problems are much smaller. In fact, an ensemble of pairwise models can often be trained much more efficiently than a single classifier, even when both have the same total number of training examples. In particular, this holds true for expensive learning algorithms whose time complexity is super-linear in the number of training examples. Thus, the key problem of LPC is typically not the training time, but instead the prediction time and storage capacity. In the following, we will recapitulate some important results concerning these problems.

7.1 Training Time

For the pairwise classification scenario, it is known that even though the number of binary classifiers is quadratic in the number of labels, their joint training time is only $O(k \cdot n)$, i.e., linear in the number labels [9]. The reason is that the individual training sets are much smaller because each of the n examples will only occur in $k-1$ different training sets. This distribution of the training effort on a large number of comparably smaller problem increases the advantage in particular for expensive classifiers with a super-linear time complexity. Obviously, these results also hold for ordered and hierarchical classification.

For multilabel classification [12], the crucial factor determining the efficiency of the approach is the average number d of labels per training example (which is often small in comparison to the total number of labels). The training complexity in this case is $O(d \cdot k \cdot n) = O(k \cdot l)$ when l is the total number of labels in the training set.

In the worst case, when all training examples are associated with a complete ranking of all labels, the training complexity is quadratic in the number of labels [22].

7.2 Prediction Time

A more interesting problem is the efficiency at prediction time. In principle, one has to query a quadratic number of classifiers in order to derive a final ranking of the classes. However, when one is only interested in classification, it is not necessary to query all classifiers in order to determine the winning class. For example, if one class has received more votes than every other class can possibly achieve in their remaining evaluations, this class can be safely predicted without querying the remaining classifiers. The QWeighted algorithm [34] tries to enforce this situation by always focusing on the class that has lost the least amount of voting mass. Experiments showed that QWeighted has an average runtime of $O(k \cdot \log k)$ instead of the $O(k^2)$ that would be required for computing the same prediction with all evaluations. Because it nevertheless produces the same predictions as regular weighted voting and thus has the same theoretical guarantees (cf. Section 6.1), it is preferred to algorithms like the pairwise DAGs [36], which only approximate the correct prediction. The algorithm can also be generalized to multilabel prediction [30] and to general ternary error-correcting output codes [35].

7.3 Memory Requirements

Even if training is quite efficient, and classification can be handled without the need to query all classifiers, we still have to store all $k \cdot (k-1)/2$ binary classifiers because each classifier will be needed for some examples (unless some labels are never predicted). For example, we have recently tackled a large-scale real-world text categorization problem, namely the annotation of the EUR-Lex database of legal documents of the European Union with labels that are taken from the EUROVOC ontology [29]. This multilabel classification task involved around 20,000 documents, each of which was labeled with about 5 out of 4000 possible labels. The key problem that had to be solved for this task was that the ensemble of almost 8,000,000 binary classifiers (perceptrons) that were trained for tackling this task could no longer be kept in main memory. The problem was solved by resorting to the dual representation of the perceptron, which reduced the total number of weights that had to be stored at the expense of a somewhat higher classification time. This made the pairwise ranking approach feasible for problems of this size. This solution, however, is only applicable to classifiers that can re-formulate their hypothesis as a linear combination of the input examples, such as perceptrons or SVMs.

For concept descriptions with varying sizes, such as rule sets or decision trees, one can expect that the learned theories for the pairwise classifiers are much smaller than the theories for larger problems such as those of a one-vs-all classifier. However, it is unclear whether these savings on individual theories can compensate for the higher number of classifiers that have to be stored. Moreover, a general solution not restricted to any specific classifier is still an open research problem. Presumably, one will have to resort to fixed approximation techniques, which allow that some classifiers are not trained at all.

8 Conclusions and Outlook

This chapter has reviewed recent work on preference learning and ranking via pairwise classification. The *learning by pairwise comparison* (LPC) paradigm has a natural motivation in the context of preference learning and goes hand in hand with the relational approach to preference modeling. Roughly speaking, a pairwise classification corresponds to a pairwise comparison between two alternatives, which in turn can be seen as a basic building block of more complex decision making procedures. Eventually, complex prediction problems can thus be reduced to the solution of a set of "simple" binary classification problems, which makes the LPC approach especially appealing from a machine learning point of view.

It was shown that LPC can be used in a quite general way to solve different types of preference learning problems. It disposes of sound theoretical foundations and has shown very strong performance in a number of empirical studies. Despite the high number of binary classifiers needed (quadratic in the number of class labels), the training time of the approach seems to be competitive with alternative approaches. Storing and querying such a large number of classifiers, however, is a more significant problem, and an active research area.

Besides, there are several other open questions and promising lines of research. From a practical point of view, for example, it would be interesting to develop a unified framework of LPC for label, instance, and object ranking. The task in label ranking, discussed in Section 3, is to learn a model that orders a fixed set of labels, where the ordering depends on the context as specified by an instance. This instance, which constitutes the input of the model, is characterized by properties (e.g., in the form of an attribute-value representation) that can be exploited by the learner, whereas the labels are mere identifiers. In object ranking, the task is to learn a model that accepts a set of objects as input and outputs a preferential order of these objects. As opposed to label ranking, the objects to be ordered are now characterized by properties, but the sought ranking is not context-dependent. Instance ranking may be viewed as a special case of both cases. In many applications, it would be desirable to combine the above two settings, i.e., to have a unified framework in which both the instances (e.g., users of a recommender system) and the labels/objects (e.g., the items to be purchased) can be described in terms of properties. While this can be achieved in a quite straightforward way for other approaches (see, e.g., Aiolli and Sperduti, this volume), it is much less obvious for LPC, mainly because the set of "labels" may become very large and change from prediction to prediction.

Another line of research concerns the prediction of preference relations more general than rankings, in particular relations which are not necessarily total (i.e., partial orders) or not strict (i.e., allow for the indifference between alternatives). As an interesting point of departure for a generalization of this type we mention recent work on the learning of *valued preference structures* [19, 18]. Instead of producing, in the first step, a single binary relation $\mathcal{R}_{\boldsymbol{x}}$ from which the final prediction (e.g., a ranking) is then derived, the idea is to predict a complete preference structure consisting of three such relations: a strict preference relation, an indifference relation, and an incomparability relation. A structure of that kind conveys much more information which can then be used, amongst other things, for predicting generalized preferences such as weak or partial orders.

Acknowledgments. This research has been supported by the German Science Foundation (DFG). We would like to thank our collaborators Klaus Brinker, Weiwei Cheng, Jens Hühn, Eneldo Loza Mencía, Sang-Hyeun Park, and Stijn Vanderlooy.

References

1. Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning*

Research, 1:113–141, 2000.
2. Cecilio Angulo, Francisco J. Ruiz, Luis González, and Juan Antonio Ortega. Multi-classification by using tri-class SVM. *Neural Processing Letters*, 23(1):89–101, 2006.
3. Ralph A. Bradley and Milton E. Terry. The rank analysis of incomplete block designs — I. The method of paired comparisons. *Biometrika*, 39:324–345, 1952.
4. Klaus Brinker, Johannes Fürnkranz, and Eyke Hüllermeier. A unified model for multilabel classification and ranking. In G. Brewka, S. Coradeschi, A. Perini, and P. Traverso, editors, *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI-06)*, pages 489–493, 2006.
5. Ofer Dekel, Chrisopher D. Manning, and Yoram Singer. Log-linear models for label ranking. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems (NIPS-03)*, pages 497–504, Cambridge, MA, 2004. MIT Press.
6. János Fodor and Marc Roubens. *Fuzzy Preference Modelling and Multicriteria Decision Support*. Kluwer Academic Publishers, 1994.
7. Eibe Frank and Mark Hall. A simple approach to ordinal classification. In L. De Raedt and P. Flach, editors, *Proceedings of the 12th European Conference on Machine Learning (ECML-01)*, pages 145–156, Freiburg, Germany, 2001. Springer-Verlag.
8. Jerome H. Friedman. Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University, Stanford, CA, 1996.
9. Johannes Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2:721–747, 2002.
10. Johannes Fürnkranz. Round robin ensembles. *Intelligent Data Analysis*, 7(5):385–404, 2003.
11. Johannes Fürnkranz and Eyke Hüllermeier. Pairwise preference learning and ranking. In N. Lavrač, D. Gamberger, H. Blockeel, and L. Todorovski, editors, *Proceedings of the 14th European Conference on Machine Learning (ECML-03)*, volume 2837 of *Lecture Notes in Artificial Intelligence*, pages 145–156, Cavtat, Croatia, 2003. Springer-Verlag.
12. Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. Multilabel classification via calibrated label ranking. *Machine Learning*, 73(2):133–153, 2008.
13. Johannes Fürnkranz, Eyke Hüllermeier, and Stijn Vanderlooy. Binary decomposition methods for multipartite ranking. In Wray L. Buntine, Marko Grobelnik, Dunja Mladenic, and John Shawe-Taylor, editors, *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD-09)*, volume Part I, pages 359–374, Bled, Slovenia, 2009. Springer-Verlag. 2009.
14. Sariel Har-Peled, Dan Roth, and Dav Zimak. Constraint classification: A new approach to multiclass classification. In N. Cesa-Bianchi, M. Numao, and R. Reischuk, editors, *Proceedings of the 13th International Conference on Algorithmic Learning Theory (ALT-02)*, pages 365–379, Lübeck, Germany, 2002. Springer.
15. Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling. In M.I. Jordan, M.J. Kearns, and S.A. Solla, editors, *Advances in Neural Information Processing Systems 10 (NIPS-97)*, pages 507–513. MIT Press, 1998.
16. Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, March 2002.

17. Jens Hühn and Eyke Hüllermeier. Is an ordinal class structure useful in classifier learning? *International Journal of Data Mining, Modelling, and Management*, 1(1):45–67, 2008.
18. Jens Hühn and Eyke Hüllermeier. FR3: A fuzzy rule learner for inducing reliable classifiers. *IEEE Transactions on Fuzzy Systems*, 17(1):138–149, 2009.
19. Eyke Hüllermeier and Klaus Brinker. Learning valued preference structures for solving classification problems. *Fuzzy Sets and Systems*, 159(18):2337–2352, 2008.
20. Eyke Hüllermeier and Johannes Fürnkranz. Comparison of ranking procedures in pairwise preference learning. In *Proceedings of the 10th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU-04)*, Perugia, Italy, 2004.
21. Eyke Hüllermeier and Johannes Fürnkranz. On predictive accuracy and risk minimization in pairwise label ranking. *Journal of Computer and System Sciences*, To appear.
22. Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172:1897–1916, 2008.
23. Eyke Hüllermeier and Stijn Vanderlooy. Combining predictions in pairwise classification: An optimal adaptive voting strategy and its relation to weighted voting. *Pattern Recognition*, 2009. To appear.
24. Thorsten 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.
25. Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. Single-layer learning revisited: A stepwise procedure for building and training a neural network. In F. Fogelman Soulié and J. Hérault, editors, *Neurocomputing: Algorithms, Architectures and Applications*, volume F68 of *NATO ASI Series*, pages 41–50. Springer-Verlag, 1990.
26. Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. Handwritten digit recognition by neural networks with single-layer training. *IEEE Transactions on Neural Networks*, 3(6):962–968, 1992.
27. Daphne Koller and Mehran Sahami. Hierarchically classifying documents using very few words. In *Proceedings of the 14th International Conference on Machine Learning (ICML-97)*, pages 170–178, Nashville, 1997.
28. Ulrich H.-G. Kreßel. Pairwise classification and support vector machines. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, chapter 15, pages 255–268. MIT Press, Cambridge, MA, 1999.
29. Eneldo Loza Mencía and Johannes Fürnkranz. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In Walter Daelemans, Bart Goethals, and Katharina Morik, editors, *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Disocvery in Databases (ECML-PKDD-2008), Part II*, pages 50–65, Antwerp, Belgium, 2008. Springer-Verlag.
30. Eneldo Loza Mencía, Sang-Hyeun Park, and Johannes Fürnkranz. Efficient voting prediction for pairwise multilabel classification. In *Proceedings of the 11th European Symposium on Artificial Neural Networks (ESANN-09)*, pages 117–122, Bruges, Belgium, 2009. d-side publications.
31. Bao-Liang Lu and Masami Ito. Task decomposition and module combination based on class relations: A modular neural network for pattern classification. *IEEE Transactions on Neural Networks*, 10(5):1244–1256, September 1999.

32. Miguel Moreira and Eddy Mayoraz. Improved pairwise coupling classification with correcting classifiers. In C. Nédellec and C. Rouveirol, editors, *Proceedings of the 10th European Conference on Machine Learning (ECML-98)*, pages 160–171, Chemnitz, Germany, 1998. Springer-Verlag.
33. Ge Hyun Nam. Ordered pairwise classification. Master's thesis, TU Darmstadt, Knowledge Engineering Group, 2007.
34. Sang-Hyeun Park and Johannes Fürnkranz. Efficient pairwise classifciation. In J. N. Kok, J. Koronacki, R. Lopez de Mantaras, S. Matwin, D. Mladenič, and A. Skowron, editors, *Proceedings of 18th European Conference on Machine Learning (ECML-07)*, pages 658–665, Warsaw, Poland, 2007. Springer-Verlag.
35. Sang-Hyeun Park and Johannes Fürnkranz. Efficient decoding of ternary error-correcting output codes for multiclass classification. In Wray L. Buntine, Marko Grobelnik, Dunja Mladenic, and John Shawe-Taylor, editors, *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD-09)*, volume Part I, pages 189–204, Bled, Slovenia, 2009. Springer-Verlag.
36. John C. Platt, Nello Cristianini, and John Shawe-Taylor. Large margin DAGs for multiclass classification. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12 (NIPS-99)*, pages 547–553. MIT Press, 2000.
37. David Price, Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. Pairwise neural network classifiers with probabilistic outputs. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7 (NIPS-94)*, pages 1109–1116. MIT Press, 1995.
38. Thomas L. Saaty. *Decision Making for Leaders: The Analytic Hierarchy Process for Decisions in a Complex World*. RWS Publications, Pittsburgh, Pennsylvania, 1999.
39. Thomas L. Saaty. Relative measurement and its generalization in decision making: Why pairwise comparisons are central in mathematics for the measurement of intangible factors – the analytic hierarchy/network process. *Revista de la Real Academia de Ciencias Exactas, Físicas y Naturales. Serie A: Matemáticas (RACSAM)*, 102(2):251–318, 2008.
40. Michael S. Schmidt and Herbert Gish. Speaker identification via support vector classifiers. In *Proceedings of the 21st IEEE International Conference Conference on Acoustics, Speech, and Signal Processing (ICASSP-96)*, pages 105–108, Atlanta, GA, 1996.
41. Jan Frederik Sima. Präferenz-Lernen für Hierarchische Multilabel Klassifikation. Master's thesis, TU Darmstadt, Knowledge Engineering Group, 2007.
42. Jan-Nikolas Sulzmann, Johannes Fürnkranz, and Eyke Hüllermeier. On pairwise naive bayes classifiers. In J. N. Kok, J. Koronacki, R. Lopez de Mantaras, S. Matwin, D. Mladenič, and A. Skowron, editors, *Proceedings of 18th European Conference on Machine Learning (ECML-07)*, pages 371–381, Warsaw, Poland, 2007. Springer-Verlag.
43. Louis Leon Thurstone. A law of comparative judgement. *Psychological Review*, 34:278–286, 1927.
44. Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. Probability estimates for multiclass classification by pairwise coupling. *Journal of Machine Learning Research*, 5(Aug):975–1005, 2004.
45. Yiming Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1:69–90, 1999.