Preference Learning: An Introduction

Johannes Fürnkranz[1] and Eyke Hüllermeier[2]

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

Abstract. This introduction gives a brief overview of the field of preference learning and, along the way, tries to establish a unified terminology. Special emphasis will be put on learning to rank, which is by now one of the most extensively studied problem tasks in preference learning and also prominently represented in this book. We propose a categorization of ranking problems into object ranking, instance ranking, and label ranking. Moreover, we introduce these scenarios in a formal way, discuss different ways in which the learning of ranking functions can be approached, and explain how the contributions collected in this book relate to this categorization. Finally, we also highlight some important applications of preference learning methods.

1 Introduction

Reasoning with preferences has been recognized as a particularly promising research direction for artificial intelligence (AI) [15]. A preference can be considered as a relaxed constraint which, if necessary, can be violated to some degree. In fact, an important advantage of a preference-based problem solving paradigm is an increased flexibility, as nicely explained in [6]:

> *"Early work in AI focused on the notion of a goal—an explicit target that must be achieved—and this paradigm is still dominant in AI problem solving. But as application domains become more complex and realistic, it is apparent that the dichotomic notion of a goal, while adequate for certain puzzles, is too crude in general. The problem is that in many contemporary application domains ... the user has little knowledge about the set of possible solutions or feasible items, and what she typically seeks is the best that's out there. But since the user does not know what is the best achievable plan or the best available document or product, she typically cannot characterize it or its properties specifically. As a result, she will end up either asking for an unachievable goal, getting no solution in response, or asking for too little, obtaining a solution that can be substantially improved."*

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 [37]. 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. Approaches relevant to this area range from approximating the utility function of a single agent on the basis of an as effective as possible question-answer process (often referred to as *preference elicitation*) to collaborative filtering where a customer's preferences are estimated from the preferences of other customers. In fact, problems of preference learning can be formalized within various settings, depending, e.g., on the underlying type of preference model or the type of information provided as an input to the learning system.

Roughly speaking, *preference learning* is about inducing predictive preference models from empirical data. In the literature on choice and decision theory, two main approaches to modeling preferences can be found, namely in terms of *utility functions* and in terms of *preference relations*. From a machine learning point of view, these two approaches give rise to two kinds of learning problems: learning utility functions and learning preference relations. The latter deviates more strongly than the former from conventional problems like classification and regression, as it involves the prediction of complex structures, such as rankings or partial order relations, rather than single values. Moreover, training input in preference learning will not, as it is usually the case in supervised learning, be offered in the form of complete examples but may comprise more general types of information, such as relative preferences or different kinds of indirect feedback and implicit preference information.

This book tries to give a comprehensive overview of the state-of-the-art in the field of preference learning. Some of its chapters are based on selected contributions to two successful workshops on this topic [29, 30], but the material is complemented with chapters that have been solicited explicitly for this book. Most notably, several survey chapters give a detailed account on ongoing research in various subfields of preference learning. Thus, we are confident that the book succeeds in giving a comprehensive survey of work on all aspects of this emerging research area.

In the remainder of this chapter, we shall briefly sketch some important branches of preference learning and, along the way, give pointers to the contributions in this volume. References to these contributions are indicated by capitalized author names, for example FÜRNKRANZ & HÜLLERMEIER.

2 Preference Learning Tasks

Among the problems in the realm of preference learning, the task of "learning to rank" has probably received the most attention in the machine learning literature in recent years. In fact, a number of different ranking problems have been introduced so far, though a commonly accepted terminology has not yet been established. In the following, we propose a unifying and hopefully clarifying terminology for the most important types of ranking problems, which will also serve as a guideline for organizing the chapters of the book. Aiolli & Sperduti give an alternative unifying framework for learning to rank from preferences.

In general, a preference learning task consists of some set of items for which preferences are known, and the task is to learn a function which predicts preferences for a new set of items, or for the same set of items in a different context. Frequently, the predicted preference relation is required to form a total order, in which case we also speak of a ranking problem. In this book, we will frequently use the term "ranking" for categorizing different types of preference learning problems, but we note that the characterization mainly depends on the form of the training data and the required predictions, and not on the fact that a total order is predicted.[3]

In the notation used in the remainder of this chapter (and throughout most of the book), our goal is to stick as much as possible to the terminology commonly used in supervised learning (classification), where a data object typically consists of an *instance* (the) input, also called predictive or independent variable in statistics) and an associated *class label* (the output, also called target or dependent variable in statistics). The former is normally denoted by $\boldsymbol{x}$, and the corresponding instance space by $\mathcal{X}$, while the output space is denoted by $\mathcal{Y}$. Instances are often represented in the form of feature vectors, which means that $\boldsymbol{x}$ is a vector

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_m) \in \mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_m \enspace .$$

We distinguish three types of ranking problems, namely label ranking, instance ranking, and object ranking, which are described in more detail in the following.

2.1 Label Ranking

In label ranking, we assume to be given an instance space $\mathcal{X}$ and a finite set of labels $\mathcal{Y} = \{y_1, y_2, \ldots, y_k\}$. The goal is to learn a "label ranker" in the form of an $\mathcal{X} \rightarrow S_{\mathcal{Y}}$ mapping, where the output space $S_{\mathcal{Y}}$ is given by the set of all total orders (permutations) of the set of labels $\mathcal{Y}$ (the notation is leaned on the

[3] Besides, one should be aware of conflicts between terminology in different fields. In the field of operations research, for example, the term "ranking" is used for arranging a set of objects in a total order, while "sorting" refers to the assignment of objects to an ordered set of categories, a problem closely related to what is called "ordered classification" in machine learning.

Given:
- a set of training instances $\{\boldsymbol{x}_\ell \mid \ell = 1, 2, \ldots, n\} \subseteq \mathcal{X}$ (each instance typically though not necessarily represented by a feature vector)
- a set of labels $\mathcal{Y} = \{y_i \mid i = 1, 2, \ldots, k\}$
- for each training instance $\boldsymbol{x}_\ell$: a set of *pairwise preferences* of the form $y_i \succ_{\boldsymbol{x}_\ell} y_j$

Find:
- a ranking function that maps any $\boldsymbol{x} \in \mathcal{X}$ to a ranking $\succ_{\boldsymbol{x}}$ of $\mathcal{Y}$ (permutation $\pi_{\boldsymbol{x}} \in \mathcal{S}_k$)

Performance measures:
- ranking error (e.g., based on rank correlation measures) comparing predicted ranking with target ranking
- position error comparing predicted ranking with a target label

Fig. 1. Label ranking

common notation S_k for the symmetric group of order k). Thus, label ranking can be seen as a generalization of conventional classification, where a complete ranking

$$y_{\pi_{\boldsymbol{x}}^{-1}(1)} \succ_{\boldsymbol{x}} y_{\pi_{\boldsymbol{x}}^{-1}(2)} \succ_{\boldsymbol{x}} \ldots \succ_{\boldsymbol{x}} y_{\pi_{\boldsymbol{x}}^{-1}(k)}$$

is associated with an instance $\boldsymbol{x}$ instead of only a single class label. Here, $\pi_{\boldsymbol{x}}$ is a permutation of $\{1, 2, \ldots, k\}$ such that $\pi_{\boldsymbol{x}}(i)$ is the position of label y_i in the ranking associated with $\boldsymbol{x}$.

The training data $\mathcal{T}$ of a label ranker typically consists of a set of pairwise preferences of the form $y_i \succ_{\boldsymbol{x}} y_j$, suggesting that, for instance $\boldsymbol{x}$, y_i is preferred to y_j. In other words, an "observation" consists of an instance $\boldsymbol{x}$ and an ordered pair of labels (y_i, y_j). The label ranking problem is summarized in Figure 1.

This learning scenario has a large number of practical applications. For example, it is relevant for the prediction of every sort of ordering of a fixed set of elements, such as the preferential order of a fixed set of products (e.g., different types of holiday apartments) based on demographic properties of a person, or the ordering of a set of genes according to their expression level (as measured by microarray analysis) based on features of their phylogenetic profile [1]. Another application scenario is meta-learning, where the task is to rank learning algorithms according to their suitability for a new dataset, based on the characteristics of this dataset [7]. Finally, every preference statement in the well-known CP-nets approach [3], a qualitative graphical representation that reflects conditional dependence and independence of preferences under a *ceteris paribus* interpretation, formally corresponds to a label ranking.

In addition, it has been observed by several authors [25, 18, 14] that many conventional learning problems, such as classification and multilabel classification, may be formulated in terms of label preferences:

- *Classification:* A single class label y_i is assigned to each example $\boldsymbol{x}_\ell$. This implicitly defines the set of preferences $\{y_i \succ_{\boldsymbol{x}_\ell} y_j \,|\, 1 \leq j \neq i \leq k\}$.
- *Multilabel classification:* Each training example $\boldsymbol{x}_\ell$ is associated with a subset $P_\ell \subseteq \mathcal{Y}$ of possible labels. This implicitly defines the set of preferences $\{y_i \succ_{\boldsymbol{x}_\ell} y_j \,|\, y_i \in L_\ell, y_j \in \mathcal{Y} \setminus P_\ell\}$.

A general framework encompassing these and other learning problems can be found in the chapter by Aiolli & Sperduti.

In each of the former scenarios, a ranking model $f : \mathcal{X} \rightarrow \mathcal{S}_k$ is learned from a subset of all possible pairwise preferences. A suitable projection may be applied to the ranking model (which outputs permutations) as a post-processing step, for example a projection to the top-rank in classification learning where only this label is relevant.

To measure the predictive performance of a label ranker, a loss function on rankings is needed. In principle, any distance or correlation measure on rankings (permutations) can be used for that purpose, for example the number of pairs of labels which are incorrectly ordered (i.e., the number of label pairs y_i and y_j such that y_i precedes y_j in the predicted ranking although y_j is actually preferred to y_i). Apart from this type of *ranking loss*, which compares a predicted ranking with a given target ranking, it is also possible to compare a predicted ranking with a single class label. For example, if this class label is the target one is looking for, then it makes sense to evaluate a predicted ranking by the position it assigns to the label; in [28], this type of error (measuring the distance of the assigned position from the top-rank) is called the *position error*.

A general survey of label ranking is given by Vembu & Gärtner. Another discussion of label ranking and related problems is given by Fürnkranz & Hüllermeier. This chapter is specifically focused on approaches which are based on the idea of *learning by pairwise comparison*, i.e., of decomposing the original problem into a set of smaller binary classification problems. Yu, Wan & Lee show how decision-tree learning algorithms like CART can be adapted to tackle label ranking learning problems by extending the concept of purity to label ranking data. Tsivtsivadze et al. show how an approach for minimizing an approximation of a ranking loss function can be extended with a semi-supervised learning technique that tries to improve predictions by minimizing the disagreement of several ranking functions which have been learned from different views of the training data.

2.2 Instance Ranking

This setting 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

Given:
- a set of training instances $\{\boldsymbol{x}_\ell \mid \ell = 1, 2, \ldots, n\} \subseteq \mathcal{X}$ (each instance typically though not necessarily represented by a feature vector)
- a set of labels $\mathcal{Y} = \{y_1, y_2, \ldots, y_k\}$ endowed with an order $y_1 < y_2 < \cdots < y_k$
- for each training instance $\boldsymbol{x}_\ell$ an associated label y_ℓ

Find:
- a ranking function that allows one to order a new set of instances $\{\boldsymbol{x}_j\}_{j=1}^t$ according to their (unknown) preference degrees

Performance measures:
- the area under the ROC-curve (AUC) in the dichotomous case ($k = 2$)
- generalizations such as the C-index in the polychotomous case ($k > 2$)

Fig. 2. Instance ranking

consists of a set $\mathcal{T}$ of labeled instances. As an example, consider the assignment of submitted papers to categories *reject*, *weak reject*, *weak accept*, and *accept*.

In contrast to the classification setting, 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).

For the case $k = 2$, this problem is well-known as the *bipartite ranking* problem. The case $k > 2$ has recently been termed *k-partite* [42] or *multipartite ranking* [19]. As an example, consider the task of a reviewer who has to rank the papers according to their quality, possibly though not necessarily with the goal of partitioning this ranking into the above four categories.

Thus, the goal of *instance ranking* —our proposal for a generic term of bipartite and multipartite ranking— is to produce a ranking in which instances from higher classes precede those from lower classes; see Figure 2 for a formalization of this task. Different types of accuracy measures have been proposed for predictions of this kind. Typically, they count the number of ranking errors, that is, the number of pairs $(\boldsymbol{x}, \boldsymbol{x}') \in X \times X$ such that $\boldsymbol{x}$ is ranked higher than $\boldsymbol{x}'$ even though the former belongs to a lower class than the latter. In the two-class case, this amounts to the well-known AUC, the area under the ROC-curve [5], which is equivalent to the Wilcoxon-Mann-Whitney statistic [47, 38]. Its generalization to multiple (ordered) classes is known as the concordance index or C-index in statistics [22].

These measures and the multipartite ranking scenario are discussed in more detail by Waegeman & De Baets. Zhang et al. discuss different methods for employing rule learning algorithms for learning bipartite rankings. This scenario has been studied for decision-tree learning, but not yet for rule learning,

Given:

- a (potentially infinite) reference set of objects $\mathcal{Z}$ (each object typically though not necessarily represented by a feature vector)
- a finite set of pairwise preferences $\boldsymbol{x}_i \succ \boldsymbol{x}_j$, $(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{Z} \times \mathcal{Z}$

Find:

- a ranking function $f(\cdot)$ that assumes as input a set of objects and returns a permutation (ranking) of this set

Performance measures:

- ranking error (e.g., based on rank correlation measures) comparing the predicted ranking with the target ranking
- top-K measures comparing the top-positions of the rankings
- retrieval measures such as precision, recall, NDCG

Fig. 3. Object ranking

where several additional problems have to be considered, such as how to combine estimates from overlapping rules into a single probability estimate.

2.3 Object Ranking

In the setting of object ranking, there is no supervision in the sense that no output or class label is associated with an object. The goal in object ranking is to learn a ranking function $f(\cdot)$ which, given a subset Z of an underlying referential set $\mathcal{Z}$ of objects as an input, produces a ranking of these objects as an output. Again, this is typically done by assigning a score to each instance and then sorting by scores.

Objects $\boldsymbol{z} \in \mathcal{Z}$ are commonly though not necessarily described in terms of an attribute-value representation. As training information, an object ranker has access to exemplary rankings or pairwise preferences of the form $\boldsymbol{z} \succ \boldsymbol{z}'$ suggesting that $\boldsymbol{z}$ should be ranked higher than $\boldsymbol{z}'$. This scenario, summarized in Figure 3, is also known as "learning to order things" [12].

As an example consider the problem of learning to rank query results of a search engine [33, 41]. 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 [34].

The performance of an object ranker can again be measured in terms of a distance function or correlation measure on rankings. In contrast to the setting of label ranking, however, the number of items to be ordered in the context of object ranking is typically much larger. Therefore, one often prefers measures that put more emphasis on the top of a ranking while paying less attention to the bottom [17]. In Web search, for example, people normally look at the top-10 results while ignoring the rest. Besides, the target is often not a "true" ranking but instead a single object or a subset of relevant objects, for example a set of documents relevant to a query. Evaluation measures especially tailored toward these types of requirements have been proposed in information retrieval. Typical examples include precision and recall as well as normalized discounted cumulative gain (NDCG) [32, 39].

An extensive survey of object ranking approaches is given by KAMISHIMA, KAZAWA & AKAHO. Subsequently, KAMISHIMA & AKAHO discuss dimensionality reduction methods for object ranking tasks, which retain the preference information as much as possible. They assume a scenario (which they call *supervised ordering*) in which total orders for multiple subsets of objects are given, and the goal is to predict an ordering of the full set of objects. DEMBCZYŃSKI et al. compare different approaches for rule-based learning of object ranking functions, namely one utility-based approach and one approach that directly learns the binary preference predicate (cf. also Section 3). An application to learning to rank documents in biomedical information retrieval is described by ARENS.

3 Preference Learning Techniques

All three of the basic learning tasks discussed in the previous section can be tackled by very similar basic techniques. In agreement with the distinction between using utility functions and binary relations for modeling preferences, two general approaches to preference learning have been proposed in the literature, the first of which is based on the idea of learning to evaluate individual alternatives by means of a utility function (Section 3.1), while the second one seeks to compare (pairs of) competing alternatives, that is, to learn one or more binary preference predicate (Section 3.2). Making sufficiently restrictive model assumptions about the structure of a preference relation, one can also try to use the data for identifying this structure (Section 3.3). Finally, local estimation techniques à la nearest neighbor can be used, which mostly leads to aggregating preferences in one way or the other (Section 3.4).

3.1 Learning Utility Functions

As mentioned previously, an established approach to modeling preferences resorts to the concept of a *utility function*. Such a function assigns an abstract degree of utility to each alternative under consideration. From a machine learning point of view, an obvious problem is to learn utility functions from given training data. Depending on the underlying utility scale, which is typically either numerical or ordinal, the problem becomes one of regression learning or ordered classification. Both problems are well-known in machine learning. However, utility functions often implicate special requirements and constraints that have to be taken into consideration such as, for example, monotonicity in certain attributes (DEMBCZYŃSKI et al.).

Besides, as mentioned earlier, training data is not necessarily given in the form of input/output pairs, i.e., alternatives (instances) together with their utility degrees, but may also consist of qualitative feedback in the form of pairwise comparisons, stating that one alternative is preferred to another one and therefore has a higher utility degree. More generally, certain types of preference information can be formalized in terms of constraints on one or more underlying utility functions. This idea forms the basis of the general framework presented by AIOLLI & SPERDUTI. Sometimes, of course, training data is less generic and more application-specific. In collaborative filtering, for example, it simply consists of an incomplete set of product ratings given by a set of users (see DE GEMMIS et al. in this volume).

In the instance and object preferences scenario, a utility function is a mapping $f : \mathcal{X} \to \mathbb{R}$ that assigns a utility degree $f(\boldsymbol{x})$ to each instance (object) $\boldsymbol{x}$ and, hence, induces a complete 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 y_i $(i = 1, \ldots, k)$; alternatively, the functions can be summarized into a single function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that maps instance/label tuples $(\boldsymbol{x}, y)$ to real-valued scores (see AIOLLI & SPERDUTI).[4] Here, $f_i(\boldsymbol{x})$ is the utility assigned to alternative y_i by instance $\boldsymbol{x}$. To obtain a ranking for $\boldsymbol{x}$, the alternatives are sorted according to these utility scores, i.e., $\succ_{\boldsymbol{x}}$ is such that $y_i \succ_{\boldsymbol{x}} y_j \Rightarrow f_i(\boldsymbol{x}) \geq f_j(\boldsymbol{x})$.

In the setting of instance ranking, the training data consists of instances for which the sought utility scores are given. Thus, the learning problem can in principle be approached by means of classification or (ordinal) regression methods. As an important difference, however, note that the goal is not to maximize classification accuracy but ranking performance. Thus, conventional learning algorithms have to be adapted correspondingly. Approaches of this kind have, e.g., been proposed in [27, 33]. In this book, WAEGEMAN & DE BAETS discuss approaches that are based on the optimization of an extension of the binary AUC to a loss function for ordinal data.

In object and label ranking, training data typically originates from a kind of indirect supervision. The learner is not given the target scores of the utility function, but constraints on the function which are derived from comparative preference information of the form "This object (or label) should have a higher utility score than that object (or label)". Thus, the challenge for the learner is to find a function which is as much as possible in agreement with these constraints.

For object ranking approaches, this idea has first been formalized by Tesauro under the name *comparison training* [44]. He 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

[4] In a sense, this alternative is not just a formally equivalent rewriting. Instead, by considering an instance/label pair as an *object*, it suggests a natural way to unify the problems of object and label ranking.

provide a real-valued evaluation of single states. Later works on learning utility function from object preference data include [46, 33, 24].

For the case of label ranking, a method for learning the functions $f_i(\cdot)$ $(i = 1, \ldots, k)$ has been proposed in the framework of *constraint classification* [25, 26]. Here, the authors proceed from linear utility functions and find a way to express a constraint of the form $f_i(\boldsymbol{x}) - f_j(\boldsymbol{x}) > 0$ (suggesting that $y_i \succ_{\boldsymbol{x}} y_j$) in the form of a binary classification example in a newly constructed, higher-dimensional space. In other words, the original label ranking problem is transformed into a single binary classification problem. This problem is solved by fitting a separating hyperplane, and from this hyperplane, the linear utility functions (identified by corresponding weight vectors) can be reconstructed. An alternative approach, so-called *log-linear models for label ranking*, has been proposed in [14]. This approach is essentially equivalent to constraint classification, as it also amounts to learning linear utility functions for all labels. Algorithmically, however, the underlying optimization problem is approached in a different way, namely by means of a boosting-based algorithm that seeks to minimize a (generalized) ranking error in an iterative way. In this book, TSIVTSIVADZE et al. present an approach for learning a utility function for label ranking via minimization of a loss function that is based on a least-squares approximation of the ranking error.

3.2 Learning Preference Relations

The key idea of this approach is to learn a binary preference relation that compares pairs of alternatives (e.g., objects or labels). The training of a model thus becomes simpler, mainly because comparative training information (suggesting that one alternative is better than another one) can be used directly instead of translating it into constraints on a (latent) utility function. On the other hand, the prediction step may become more difficult, since a binary preference relation learned from data is not necessarily consistent in the sense of being transitive and, therefore, does normally not define a ranking in a unique way.

Binary preference relations can be turned into a ranking by finding a ranking that is maximally consistent with the corresponding pairwise preferences. The difficulty of this optimization problem depends on the concrete criterion, though many natural objectives (e.g., minimizing the number of object pairs whose ranks are in conflict with their pairwise preference) lead to NP-hard problems [12]. Fortunately, efficient techniques such as simple voting (known as the Borda count procedure in social choice theory) often deliver good approximations, sometimes even with provable guarantees [13]. Of course, one can also derive other, possibly more complex preference structures from a preference relation, for example weak instead of strict linear orders. In [45], a linear order with ties (indifference between two alternatives) is called a *bucket order* (a total order of "buckets", where each bucket corresponds to an equivalence class), and a method is proposed to find an order of this type which is maximally consistent with the data.

For object ranking problems, the relational approach has been pursued in [12]. The authors propose to solve object ranking problems by learning a binary preference predicate $Q(\boldsymbol{x}, \boldsymbol{x}')$, which predicts whether $\boldsymbol{x}$ is preferred to $\boldsymbol{x}'$ or vice versa. This predicate is trained on the basis of exemplary preferences of the form $\boldsymbol{x} \succ \boldsymbol{x}'$. A final ordering is found in a second phase by deriving (an approximation of) a ranking that is maximally consistent with these predictions. Dembczyński et al. discuss this setting for rule learning and propose to combine the predictions using the Net Flow score proposed in [4]. They also compare this setting with an alternative approach that directly learns a utility function based on the preferences and monotonicity constraints.

For label ranking problems, the pairwise approach has been introduced by [18, 31], where it is referred to as *ranking by pairwise comparison*. The key idea is to learn, for each pair of labels (y_i, y_j), a binary predicate $\mathcal{M}_{i,j}(\boldsymbol{x})$ that predicts whether $y_i \succ_{\boldsymbol{x}} y_j$ or $y_j \succ_{\boldsymbol{x}} y_i$ for an input $\boldsymbol{x}$. A label ranking is then derived from these pairwise preferences via weighted voting (generalized Borda counting).

Pairwise learning techniques for instance ranking have been proposed in [19]. More specifically, two approaches were developed and compared in that paper, one which trains binary models $\mathcal{M}_{i,j}$, one for each pair of labels y_i and y_j, and another one that trains models $\mathcal{M}_i$ $(i = 1, \ldots, k-1)$ to separate classes $y_1, \ldots, y_i$ from classes $y_{i+1}, \ldots, y_k$. Given a new query instance $\boldsymbol{x}$, both approaches submit this instance to all models that have been learned and aggregate the corresponding predictions into an overall score. A set of instances is then ranked according these scores.

An overview of work on learning binary preference relations for label and instance ranking is given by Fürnkranz & Hüllermeier.

3.3 Model-based Preference Learning

Another approach to learning ranking functions is to proceed from specific model assumptions, that is, assumptions about the structure of the preference relations. This approach is less generic than the previous ones, as it strongly depends on the concrete assumptions made.

An example is the assumption that the target ranking of a set of objects described in terms of multiple attributes can be represented as a *lexicographic order*. Yaman et al. address the learning of lexicographic orders in the context of object ranking. From a machine learning point of view, assumptions of the above type can be seen as an inductive bias restricting the hypothesis space. Provided the bias is correct, this is clearly an advantage, as it may simplify the learning problem. In the case of lexicographic orders, for example, a complete ranking of all objects is uniquely identified by a total order of the attributes plus a total order of each of the attribute domains. For example, suppose objects to be described in terms of (only) four binary attributes. Thus, there are $2^4 = 16$ objects and hence $16! \approx 2 \cdot 10^{13}$ rankings in total. However, only $(2^4) \cdot 4! = 384$ of these rankings can be expressed in terms of a lexicographic order.

Needless to say, the bias induced by the assumption of a lexicographic order is very strong and will be rarely justified in practical applications. In particular, preferences on individual attribute values will normally not be independent of each other. For example, red wine might be preferred as a beverage if the main dish is meat, while white wine might be preferred in the case of fish. As mentioned earlier, CP-nets [3] offer a language for expressing preferences on the values of single attributes and provide a suitable formalism for modeling dependencies of this type. A compact representation of a complex preference (partial order) relation is achieved by making use of conditional independence relations between such preferences (which are interpreted in terms of a *ceteris paribus* semantics), in much the same way as Bayesian networks reduce complexity of probability models by exploiting conditional independence between random variables. The CP-net itself is a graphical representation of these (in)dependencies, and each node (belonging to a single variable) is associated with a function that assigns a preference relation on the values of that attribute to each combination of values of the parent attributes. In this volume, Chevaleyre et al. discuss the learnability of CP-networks, both in a passive and an active learning scenario.

If a ranking function is defined implicitly via an underlying utility (scoring) function, the latter is normally also restricted by certain model assumptions. For example, the approaches outlined in Section 3.1 make use of linear functions to represent scores, although mostly for algorithmic reasons. There are other approaches in which the choice of the underlying utility function is more intrinsically motivated and addressed in a more explicit way. For example, Giesen et al. describe an approach for conjoint analysis, also called multi-attribute compositional models, which originated in mathematical psychology and is nowadays widely used in the social sciences and operations research. Proceeding from the description of objects in terms of a set of attributes, they assume that an underlying utility function can be decomposed into a linear sum of individual utility functions, one for each attribute. These utility functions can then be learned efficiently from the data.

Like in the case of lexicographic orders, this model assumption is obviously quite restrictive. Torra presents a complementary and more general approach. Starting with a discussion of general properties that aggregation operators for attribute-based utility functions should fulfill, he surveys different approaches for learning a complex aggregation operator in the form of a non-additive integral.

3.4 Local Aggregation of Preferences

Yet another alternative is to resort to the idea of local estimation techniques as prominently represented, for example, by the nearest neighbor estimation principle: Considering the rankings observed in similar situations as representative, a ranking for the current situation is estimated on the basis of these "neighbored" rankings, typically using an averaging-like aggregation operator. This approach is in a sense orthogonal to the previous model-based one, as it is very flexible and typically comes with no specific model assumption (except the regularity assumption underlying the nearest neighbor inference principle).

For label ranking, the nearest neighbor (instance-based learning) approach was first used in [9, 10]. Roughly speaking, the idea is to identify the query's k nearest neighbors in the instance space $\mathcal{X}$, and then to combine the corresponding rankings into a prediction using suitable aggregation techniques. In [11], this approach was developed in a theoretically more sophisticated way, realizing the aggregation step in the form of maximum likelihood estimation based on a statistical model for rank data. Besides, this approach is also able to handle the more general case in which the rankings of the neighbored instances are only partially known.

In the same paper, the authors propose to use this estimation principle for decision tree learning, too, namely for aggregating the label rankings associated with the instances grouped in a leaf node of the tree. Indeed, decision tree learning can also be seen as a kind of local learning method, namely as a piecewise constant approximation of the target function.[5] In this book, Yu, Wan & Lee propose a similar method. They grow a decision tree and propose two splitting measures for label ranking data. The rankings that are predicted at the leaves of the trees are derived by aggregating the rankings of all training examples arriving at this leaf.

Aggregation techniques are also used for other types of preference learning problems, including object ranking. For example, assume the rankings of several subsets X_i of a reference set $\mathcal{X}$ to be given. The learning task is to combine these rankings into a complete ranking of all objects in $\mathcal{X}$. A practical application of this setting occurs, e.g., in information retrieval, when different rankings of search results originating from different search engines should be combined into an overall ranking of all retrieved pages [16]. Amongst other things, the learning problem may involve the determination of suitable weights for the information sources (search engines), reflecting their performance or agreement with the preferences of the user [35]. Another example is the ranking of sports teams or players, where individual tournament results with varying numbers of participants have to be combined into an overall ranking [2], or where different rankings by different judges have to be aggregated into an overall ranking of the participants of a competition [23].

4 Applications of Preference Learning

Preference learning problems in general, and ranking problems in particular, arise quite naturally in many application areas. For example, a search engine should rank Web pages according to a user's preferences. Likewise, cars can be ranked according to a customer's preferences on characteristics that discriminate different models. Another example is the ranking of possible keywords according to their relevance for an article. A few more examples have already been given in this article, and many more could be found.

In particular in the field of information retrieval, ranking applications occur quite naturally. Two particularly interesting problems are learning to rank the results of a query to a search engine, and learning to rank possible recommendations for new products. We will briefly discuss research in these areas below.

[5] More general approximations can be realized by labeling a leaf node with a non-constant function, for example a linear function in regression learning.

4.1 Learning to Rank Search Results

A widely studied preference learning problem is the ranking of retrieval results of a search engine. Roughly, the problem is the following: given a query $\mathbf{q}$ and a set of documents $\mathcal{D}$, find a ranking of the documents in $\mathcal{D}$ that corresponds to their relevance with respect to $\mathbf{q}$. This ranking is based on an unknown preference relation $\succ_{\mathbf{q}}$, which ought to be learned from user feedback on past rankings of retrieval results for different queries. An elaborate survey of current research in this area can be found in [36].

Current research focuses particularly on suitable ways of characterizing the queries that allow one to transfer the ranking from one query to another [20, 40]. In some sense, a query may be considered as a context for a ranking, and the task is to learn a function that allows one to transfer the ranking from one context to the other. Much of the research is conducted on the LETOR (LEarning TO Rank for information retrieval) collection, a package of datasets containing a wide variety of queries with user feedback in several domains.[6]

Users can provide explicit feedback by labeling the retrieved pages with their degree of relevance. However, users are only willing to do this for a limited number of pages. It would be better if feedback could be collected in a way that is transparent to the user. RADLINSKI & JOACHIMS discuss a variety of techniques that allow one to collect the user feedback implicitly via their clicking behavior. Alternatively, ARENS proposes the use of active learning techniques which help minimizing the burden on the user by a careful automatic selection of suitable training examples for the ranking algorithm. He illustrates his technique in an application which learns a ranking function for the PubMed search engine for the MEDLINE database of biomedical literature.

4.2 Recommender Systems

Nowadays, *recommender systems* [43] are frequently used by on-line stores to recommend products to their customers. Such systems typically store a data table with products over users, which records the degree of preference of a user for this product. A customer can provide this preference degree explicitly by giving some sort of feedback (e.g., by assigning a rating to a movie) or implicitly (e.g., by buying the DVD of the movie). The elegant idea of collaborative filtering systems [21] is that recommendations can be based on user similarity, and that user similarity can in turn be defined by the similarity of their recommendations. Alternatively, recommender systems can also be based on item similarities, which are defined via the recommendations of the users that recommended the items in question. Yet other approaches try to learn models that capture the preference information contained in the matrix. A very good comparison of these approaches can be found in [8].

In this book, DE GEMMIS et al. given an extensive survey of recommender systems. Subsequently, KARATZOGLOU & WEIMER describe the use of matrix factorization methods for compressing the information contained in the

[6] `http://research.microsoft.com/en-us/um/beijing/projects/letor/`

user/product matrix into a product of two lower-dimensional matrices. The dimensions over which the product is computed may be viewed as hidden concepts which can be used to categorize the interests of a user. Interestingly, only very few (in the order of 10) such concepts are enough for a sufficiently accurate representation of large numbers of users and products. Finally, Bellogin et al. describe an approach that uses decision tree learning for identifying features of recommendation models that influence the quality of the predicted preference ranking.

5 Conclusions

In this introductory chapter, we have tried to give an overview of different approaches to preference learning, categorized by the learning task (label, instance, or object ranking) and the learning technique (learning utility functions, learning binary preference relations, learning preference models having a specific structure, or using local estimation and preference aggregating methods). In principle, all task/technique combinations are conceivable and can be found in the literature. We also highlighted important application areas, in particular in ranking search results and product recommendations.

Throughout the chapter, pointers to the remaining articles in this book were given. They could be characterized along a variety of dimensions, including all of those mentioned above. We have adopted a grouping that more or less corresponds to the subsections in this survey. As the categorization is multidimensional, most articles fit into several of these categories, and, in some cases, the choice was not entirely clear. This is why we do not have separate parts on learning of a utility function or a binary preference relation, for example. In fact, almost all approaches presented in this book follow either of the two approaches.

Acknowledgments

This research has been supported by the German Science Foundation (DFG).

References

1. Rajarajeswari Balasubramaniyan, Eyke Hüllermeier, Nils Weskamp, and Jörg Kämper. Clustering of gene expression data using a local shape-based similarity measure. *Bioinformatics*, 21(7):1069–1077, 2005.
2. Adriana Birlutiu and Tom Heskes. Expectation propagation for rating players in sports competitions. In J. N. Kok, J. Koronacki, R. López de Mántaras, S. Matwin, D. Mladenić, and A. Skowron, editors, *Proceedings of the 11th European Symposium on Principles of Knowledge Discovery in Databases (PKDD-07)*, pages 374–381, Warsaw, Poland, 2007. Springer-Verlag.
3. Craig Boutilier, Ronen Brafman, Carmel Domshlak, Holger Hoos, and David Poole. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21:135–191, 2004.

4. Denis Bouyssou. Ranking methods based on valued preference relations: A characterization of the net flow method. *European Journal of Operational Research*, 60(1):61–67, 1992.
5. Andrew P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.
6. Ronen I. Brafman. Preferences, planning and control. In G. Brewka and J. Lang, editors, *Proceedings of the 11th Conference on Principles of Knowledge Representation and Reasoning (KR-08)*, pages 2–5, Sydney, Australia, 2008. AAAI Press.
7. Pavel B. Brazdil, Carlos Soares, and J. P. da Costa. Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. *Machine Learning*, 50(3):251–277, March 2003.
8. John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In G.F. Cooper and S. Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 43–52, Madison, WI, 1998. Morgan Kaufmann.
9. Klaus Brinker and Eyke Hüllermeier. Case-based label ranking. In J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, editors, *Proceedings of the 17th European Conference on Machine Learning (ECML-06)*, pages 566–573, Berlin, Germany, 2006. Springer-Verlag.
10. Klaus Brinker and Eyke Hüllermeier. Case-based multilabel ranking. In M. M. Veloso, editor, *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 702–707, Hyderabad, India, 2007.
11. Weiwei Cheng, Jens Hühn, and Eyke Hüllermeier. Decision tree and instance-based learning for label ranking. In A. Danyluk, L. Bottou, and M. L. Littman, editors, *Proceedings of the 26th International Conference on Machine Learning (ICML-09)*, Montreal, Canada, 2009.
12. William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.
13. Don Coppersmith, Lisa Fleischer, and Atri Rudra. Ordering by weighted number of wins gives a good ranking for weighted tournaments. In *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA-06)*, pages 776–782, 2006.
14. Ofer Dekel, Chrisopher D. Manning, and Yoram Singer. Log-linear models for label ranking. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems (NIPS-03)*, pages 497–504, Cambridge, MA, 2004. MIT Press.
15. Jon Doyle. Prospects for preferences. *Computational Intelligence*, 20(2):111–136, 2004.
16. Cynthia Dwork, Ravi Kumara, Moni Naor, and D. Sivakumar. Rank aggregation methods for the Web. In *Proceedings of the 10th International World Wide Web Conference (WWW-01)*, pages 613–622, Hong Kong, China, 2001. ACM.
17. Ronald Fagin, Ravi Kumar, and D. Sivakumar. Comparing top k lists. *SIAM Journal of Discrete Mathematics*, 17(1):134–160, 2003.
18. Johannes Fürnkranz and Eyke Hüllermeier. Pairwise preference learning and ranking. In Nada Lavrač, D. Gamberger, Hendrik 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.
19. 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.

20. Xiubo Geng, Tie-Yan Liu, Tao Qin, Andrew Arnold, Hang Li, and Heung-Yeung Shum. Query dependent ranking using k-nearest neighbor. In S.-H. Myaeng, D. W. Oard, F. Sebastiani, T.-S. Chua, and M.-K. Leong, editors, *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-08)*, pages 115–122, Singapore, 2008. ACM.
21. David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave and information tapestry. *Communications of the ACM*, 35(12):61–70, December 1992.
22. Mithat Gönen and Glenn Heller. Concordance probability and discriminatory power in proportional hazards regression. *Biometrika*, 92(4):965–970, 2005.
23. Stephen Gordon and Michel Truchon. Social choice, optimal inference and figure skating. *Social Choice and Welfare*, 30(2):265–284, 2008.
24. Peter Haddawy, Vu Ha, Angelo Restificar, Benjamin Geisler, and John Miyamoto. Preference elicitation via theory refinement. *Journal of Machine Learning Research*, 4:317–337, 2003.
25. 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.
26. Sariel Har-Peled, Dan Roth, and Dav Zimak. Constraint classification for multiclass classification and ranking. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15 (NIPS-02)*, pages 785–792, 2003.
27. Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Large margin rank boundaries for ordinal regression. In P. J. Bartlett, B. Schölkopf, D. Schuurmans, and A. J. Smola, editors, *Advances in Large Margin Classifiers*, pages 115–132. MIT Press, 2000.
28. Eyke Hüllermeier and Johannes Fürnkranz. Learning label preferences: Ranking error versus position error. In *Advances in Intelligent Data Analysis: Proceedings of the 6th International Symposium (IDA-05)*, pages 180–191, Madrid, Spain, 2005. Springer-Verlag.
29. Eyke Hüllermeier and Johannes Fürnkranz, editors. *Proceedings of the ECML/PKDD-08 Workshop on Preference Learning*, Antwerp, Belgium, 2008.
30. Eyke Hüllermeier and Johannes Fürnkranz, editors. *Proceedings of the ECML/PKDD-09 Workshop on Preference Learning*, Bled, Slovenia, 2009.
31. Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172:1897–1916, 2008.
32. K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.
33. 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.
34. Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri 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)*, pages 154–161, 2005.

35. Guy Lebanon and John Lafferty. Cranking: Combining rankings using conditional probability models on permutations. In C. Sammut and A. Hoffmann, editors, *Proceedings of the 19th International Conference on Machine Learning (ICML-02)*, pages 363–370, Sydney, Australia, 2002.
36. Tie-Yan Lu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.
37. R. Duncan Luce and Howard Raiffa. *Games and Decisions: Introduction and Critical Survey.* Wiley, New York, NY, 1957.
38. Henry Berthold Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 18(50):50–60, 1947.
39. Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval.* Cambridge University Press, 2008.
40. Weijian Ni, Yalou Huang, and Maoqiang Xie. A query dependent approach to learning to rank for information retrieval. In *Proceedings of the 9thh International Conference on Web-Age Information Management (WAIM-08)*, pages 262–269, Zhangjiajie, China, 2008. IEEE.
41. Filip Radlinski and Thorsten Joachims. Learning to rank from implicit feedback. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD-05)*, pages 239–248, 2005.
42. Shyamsundar Rajaram and Shivani Agarwal. Generalization bounds for k-partite ranking. In S. Agarwal, C. Cortes, and R. Herbrich, editors, *Proceedings of the NIPS 2005 Workshop on Learning to Rank*, pages 28–23, Whistler, BC, Canada, 2005.
43. Paul Resnick and Hal R. Varian. Special issue on recommender systems. *Communications of the ACM*, 40(3), 1997.
44. 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.
45. Antti Ukkonen, Kai Puolamäki, Aristides Gionis, and Heikki Mannila. A randomized approximation algorithm for computing bucket orders. *Information Processing Letters*, 109:356–359, 2009.
46. Jun Wang. Artificial neural networks versus natural neural networks: A connectionist paradigm for preference assessment. *Decision Support Systems*, 11:415–429, 1994.
47. Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1:80–83, 1945.