Recommended Reading

Briand, L., El Emam, K., Surmann, D., & Wieczorek, I. (1999). An assessment and comparison of common software cost estimation modeling techniques. In *Proceedings of 21st international conference on software engineering* (pp. 313–322).

Cohen, W., & Devanbu, P. (1999). Automatically exploring hypotheses about fault prediction: A comparative study of inductive logic programming methods. *International Journal of Software Engineering and Knowledge Engineering*, *9*(5), 519–546.

Dolado, J. J. (2000). A validation of the component-based method for software size estimation. *IEEE Transactions on Software Engineering*, *26*(10), 1006–1021.

El Emam, K., Benlarbi, S., Goel, N., & Rai, S. (2001). Comparing case-based reasoning classifiers for predicting high risk software components. *Journal of Systems and Software*, *55*(3), 301–320.

Esteva, J. C. (1990). Learning to recognize reusable software modules using an inductive classification system. In *Proceedings of the fifth Jerusalem conference on information technology* (pp. 278–285).

Evett, M., & Khoshgoftar, T. (1998). GP-based software quality prediction. In *Proceedings of the third annual conference on genetic programming* (pp. 60–65).

Fenton, N. E., & Pfleeger, S. L. (1998). *Software metrics: A rigorous and practical approach* (2nd ed.). Boston: PWS.

Fenton, N., & Neil, M. (1999). A critique of software defect prediction models. *IEEE Transactions on Software Engineering*, *25*(5), 675–689.

Ganek, A. G., & Corbi T. A. (2003). The dawning of autonomic computing era. *IBM Systems Journal*, *42*(1), 5–18.

Grosser, D., Sahraoui, H. A., & Valtchev, P. (2002). Predicting software stability using case-based reasoning. In *Proceedings of 17th IEEE international conference on automated software engineering (ASE)* (pp. 295–298).

Khoshgoftaar, T., Allen, E., Hudepohl, J., & Aud, S. (1997). Applications of neural networks to software quality modeling of a very large telecommunications system. *IEEE Transactions on Neural Networks*, *8*(4), 902–909.

Ling, C., Sheng, V., Bruckhaus, T., & Madhavji, N. (2006). Maximum profit mining and its application in software development. In *Proceedings of the 12th ACM international conference on knowledge discovery and data mining (SIGKDD)* (pp. 929–934).

Mao, Y., Sahraoui, H., & Lounis, H. (1998). Reusability hypothesis verification using machine learning techniques: A case study. In *Proceedings of the 13th IEEE international conference on automated software engineering* (pp. 84–93).

Oliveira, A. (2006). Estimation of software project effort with support vector regression. *Neurocomputing*, *69*(13–15), 1749–1753.

Padberg, F., Ragg, T., & Schoknecht, R. (2004). Using machine learning for estimating the defect content after an inspection. *IEEE Transactions on Software Engineering*, *30*(1), 17–28.

Pai, P. F., & Hong, W. C. (2006). Software reliability forecasting by support vector machines with simulated annealing algorithms. *Journal of Systems and Software*, *79*(6), 747–755.

Pendharkar, P. C. (2004). An exploratory study of object-oriented software component size determinants and the application of regression tree forecasting models. *Information and Management*, *42*(1), 61–73.

Pfleeger, S. L., & Atlee J. M. (2003). *Software engineering: Theory and practice*. Upper Saddle River, NJ: Prentice-Hall.

Quah, T. S., & Thwin, M. M. T. (2003). *Application of neural networks for software quality prediction using object-oriented metrics*. In *Proceedings of international conference on software maintenance* (pp. 22–26).

Regolin, E. N., de Souza, G. A., Pozo, A. R. T., & Vergilio, S. R. (2003). *Exploring machine learning techniques for software size estimation*. In *Proceedings of the 23rd international conference of the Chilean computer science society (SCCC)* (pp. 130–136).

Sayyad Shirabad, J., Lethbridge, T. C., & Matwin, S. (2007). Modeling relevance relations using machine learning techniques. In J. Tsai & D. Zhang (Eds.), *Advances in machine learning applications in software engineering* (pp. 168–207). IGI.

Seliya, N. & Khoshgoftaar, T. M. (2007). Software quality estimation with limited fault data: a semi-supervised learning perspective. *Software Quality Journal*, *15*(3), 327–344.

Shepperd, M., & Schofield, C. (1997). Estimating software project effort using analogies. *IEEE Transactions on Software Engineering*, *23*(11), 736–743.

Sitte, R. (1999). Comparison of software-reliability-growth predictions: neural networks vs parametric-recalibration. *IEEE Transactions on Reliability*, *48*(3), 285–291.

Xing, F., Guo, P., & Lyu, M. R. (2005). A novel method for early software quality prediction based on support vector machine. In *Proceedings of IEEE international conference on software reliability engineering* (pp. 213–222).

Zhang, Du., & Tsai, J. P. (2003). Machine learning and software engineering. *Software Quality Journal*, *11*(2), 87–119.

Preference Learning

Johannes Fürnkranz[1], Eyke Hüllermeier[2]
[1]TU Darmstadt
[2]Philipps-Universität Marburg

Synonyms

Learning from preferences

Definition

Preference learning refers to the task of learning to predict an order relation on a collection of objects (alternatives). In the training phase, preference learning algorithms have access to examples for which the sought order relation is (partially) known. Depending on the formal modeling of the preference context and the alternatives to be ordered, one can distinguish between *object* ranking problems and *label* ranking problems. Both types of problems can be approached in two fundamentally different ways, either by modeling the binary preference relation directly, or by inducing this relation indirectly via an underlying (latent) utility function.

Motivation and Background

Preference information plays a key role in automated decision making and appears in various guises in Artificial Intelligence (AI) research, notably in fields such as agents, non-monotonic reasoning, constraint satisfaction, planning, and qualitative decision theory (Doyle, 2004). Preferences provide a means for specifying desires in a declarative way, which is a point of critical importance for AI. In fact, considering AI's paradigm of a rationally acting (decision-theoretic) agent, the behavior of such an agent has to be driven by an underlying preference model, and an agent recommending decisions or acting on behalf of a user should clearly reflect that user's preferences. Therefore, the formal modeling of preferences can be considered an essential aspect of autonomous agent design.

Drawing on past research on knowledge representation and reasoning, AI offers qualitative and symbolic methods for *modeling* and *processing* preferences that can reasonably complement standard approaches from economic decision theory, namely numerical *utility functions* and binary *preference relations*.

In practice, preference modeling can still become a rather cumbersome task if it must be done by hand. This is an important motivation for preference *learning*, which is meant to support and partly automatize the design of preference models. Roughly speaking, preference learning is concerned with the automated acquisition of preference models from data, that is, data from which (possibly uncertain) preference information can be deduced in a direct or indirect way.

Computerized methods for revealing the preferences of individuals (users) are useful not only in AI, but also in many related fields, notably in areas such as information retrieval, information systems, and e-commerce, where an increasing trend toward *personalization* of products and services can be recognized.Correspondingly, a number of methods and tools, such as recommender systems and collaborative filtering, have been proposed in the recent literature, which could in principle be subsumed under the heading of preference learning. In fact, one should realize that preference learning is a relatively recent and emerging topic. A first attempt for setting a common framework in this area can be found in Fürnkranz and Hüllermeier (2010). In this article, we shall therefore focus on two particular learning tasks that have been studied in the realm of machine learning and can be considered as extensions of classical machine learning problems.

Before proceeding, we introduce some basic notation that will be used later on. A weak preference relation $\succeq$ on a set $\mathcal{A}$ is a reflexive and transitive binary relation. Such a relation induces a strict preference $\succ$ and an indifference relation $\sim$ as follows: $a \succ b$ iff $(a \succeq b)$ and $(b \not\succeq a)$; moreover, $a \sim b$ iff $(a \succeq b)$ and $(b \succeq a)$. In agreement with our preference semantics, we shall interpret $a \succeq b$ as "alternative a is at least as good as alternative b." Let us note, however, that the term "preference" should not be taken literally and instead always be interpreted in a wide sense as a kind of order relation. Thus, $a \succ b$ may indeed mean that alternative a is more liked by a person than b, but also, e.g., that a is an algorithm that outperforms b on a certain problem, or that a is a student finishing her studies before another student b.

Subsequently, we shall focus on an especially simple type of preference structure, namely *total strict orders* or *rankings*, that is, relations $\succ$ which are total, irreflexive, and transitive. If $\mathcal{A}$ is a finite set $\{a_1, \ldots, a_m\}$, a ranking of $\mathcal{A}$ can be identified with a permutation τ of $\{1, \ldots, m\}$, as there is a unique permutation τ such that $a_i \succ a_j$ if and only if $\tau(i) < \tau(j)$ ($\tau(i)$ is the position of a_i in the ranking). We shall denote the class of all permutations of $\{1, \ldots, m\}$ by $\mathcal{S}_m$. Moreover, by abuse of notation, we shall sometimes employ the terms "ranking" and "permutation" synonymously.

Structure of the Learning System

As mentioned before, a considerable number of diverse approaches have been proposed under terms like ranking and preference learning. In the following, we shall distinguish between *object ranking problems*, where the task is to order subsets of objects, and *label ranking problems*, where the task is to assign a permutation of a fixed set of labels to a given instance. An important difference between these problems concerns the formal representation of the preference context and the alternatives to be ordered: In object ranking, the objects themselves are characterized by properties, typically in terms of an attribute-value representation. Thus, the ranking model can refer to *properties of the alternatives* and can therefore be applied to arbitrary sets of such alternatives. In

label ranking, the alternatives to be ranked are labels as in classification learning, i.e., mere identifiers without associated properties. Instead, the ranking context is characterized in terms of a (ranking) instance from a given instance space, and the task of the model is to rank alternatives depending on *properties of the context*. Thus, the context may now change (as opposed to object ranking, where it is implicitly fixed) but the objects to be ranked remain the same. Or, stated differently, object ranking is the problem to rank varying sets of objects under invariant preferences, whereas label ranking is the problem to rank an invariant set of objects under varying preferences.

For both problem types, there are two principal ways to approach them. One possibility is to learn a *utility function* that induces the sought ranking by *evaluating individual objects*. The alternative is to *compare pairs of objects*, that is, to learn a *binary preference relation*.

Note that the first approach implicitly assumes an underlying total order relation, since numerical (or at least totally ordered) utility scores enforce the comparability of alternatives. The second approach is more general in this regard, as it also allows for partial order relations. On the other hand, this approach may lead to complications if the target is indeed a total order, since a set of *hypothetical* binary preferences induced from empirical data is not necessarily transitive.

Learning from Object Preferences

Given:

- A (potentially infinite) set $\mathcal{X}$ of objects (each object typically represented by a feature vector)
- A finite set of pairwise preferences $x_i \succ x_j$, $(x_i, x_j) \in \mathcal{X} \times \mathcal{X}$

Find:

- A ranking function $r(\cdot)$ that, given a set of objects $\mathcal{O} \subseteq \mathcal{X}$ as input, returns a permutation (ranking) of these objects

The most frequently studied problem in learning from preferences is to induce a *ranking function* $r(\cdot)$ that is able to order any subset $\mathcal{O}$ of an underlying class $\mathcal{X}$ of objects. That is, $r(\cdot)$ assumes as input a subset $\mathcal{O} = \{x_1, \ldots, x_n\} \subseteq \mathcal{X}$ of objects and returns as output a permutation τ of $\{1, \ldots, n\}$. The interpretation of this permutation is that object x_i is preferred to x_j whenever $\tau(i) < \tau(j)$. The objects themselves are typically characterized by a finite set of features as in conventional attribute-value learning. The training data consists of a set of exemplary pairwise preferences. A survey of object ranking approaches can be found in Kamishima et al. (2010).

Note that, in order to evaluate the predictive performance of a ranking algorithm, an accuracy measure is needed that compares a predicted ranking with a given reference. To this end, one can refer, for example, to so-called ▶rank correlation measures that have been proposed in statistics. In the context of ranking, such measures play the role of, say, the classification rate in classification learning.

As an example of object ranking consider the problem of learning to rank query results of a search engine (Joachims, 2002). The training information could be 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 (Radlinski et al., 2010).

Learning from Label Preferences

Given:

- A set of training instances $\{x_k \mid k = 1, \ldots, n\} \subseteq \mathcal{X}$ (each instance typically represented by a feature vector)
- A set of labels $\mathcal{L} = \{\lambda_i \mid i = 1, \ldots, m\}$
- For each training instance x_k: a set of associated *pairwise preferences* of the form $\lambda_i \succ_{x_k} \lambda_j$

Find:

- A ranking function in the form of an $\mathcal{X} \longrightarrow \mathcal{S}_m$ mapping that assigns a ranking (permutation) $\succ_x$ of $\mathcal{L}$ to every $x \in \mathcal{X}$

In this learning scenario, the problem is to predict, for any instance x (e.g., a person) from an instance space $\mathcal{X}$, a preference relation (ranking) $\succ_x \subseteq \mathcal{L} \times \mathcal{L}$ among a finite set $\mathcal{L} = \{\lambda_1, \ldots, \lambda_m\}$ of labels or alternatives, where $\lambda_i \succ_x \lambda_j$ means that instance x prefers the label

λ_i to the label λ_j. More specifically, as we are especially interested in the case where $\succ_x$ is a total strict order, the problem is to predict a permutation of $\mathcal{L}$. The training information consists of a set of instances for which (partial) knowledge about the associated preference relation is available. More precisely, each training instance x is associated with a subset of all pairwise preferences. Thus, despite the assumption of an underlying ("true") target ranking, the training data is not expected to provide full information about such rankings. Besides, in order to increase the practical usefulness of the approach, learning algorithms should even allow for inconsistencies, such as pairwise preferences which are conflicting due to observation errors.

The above formulation follows (Hüllermeier et al. 2008), similar formalizations have been proposed independently by several authors (Dekel et al., 2004; Fürnkranz and Hüllermeier, 2003; Har-Peled et al., 2002). A survey can be found in Vembu and Gärtner (2010). Aiolli and Sperduti (2010) proposed an interesting generalization of this framework that allows one to specify both qualitative and quantitative preference constraints on an underlying utility function. In addition to comparing pairs of alternatives, it is possible to specify constraints of the form $\lambda_i \succeq_x t$, which means that the utility score of alternative x reaches the numerical threshold t.

Label ranking contributes to the general trend of extending machine learning methods to complex and structured output spaces (Fürnkranz and Hüllermeier, 2010; Tsochantaridis et al., 2004). Moreover, label ranking can be viewed as a generalization of several standard learning problems. In particular, the following well-known problems are special cases of learning label preferences:

- ►*Classification*: A single class label λ_i is assigned to each example x_k. This is equivalent to the set of preferences $\{\lambda_i \succ_{x_k} \lambda_j \,|\, 1 \leq j \neq i \leq m\}$.
- ►*Multi-label classification*: Each training example x_k is associated with a subset $L_k \subseteq \mathcal{L}$ of possible labels. This is equivalent to the set of preferences $\{\lambda_i \succ_{x_k} \lambda_j \,|\, \lambda_i \in L_k, \lambda_j \in \mathcal{L} \setminus L_k\}$.

In each of the former scenarios, the sought prediction can be obtained by post-processing the output of a ranking model $f : \mathcal{X} \longrightarrow \mathcal{S}_m$ in a suitable way. For example, in classification learning, where only a single label is requested, it suffices to project a label ranking to the top-ranked label.

Applications of this general framework can be found in various fields, for example in marketing research; here, one might be interested in discovering dependencies between properties of clients and their preferences for products. 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. Moreover, every preference statement in the well-known CP-nets approach (Boutilier et al., 2004), a qualitative graphical representation that reflects conditional dependence and independence of preferences under a *ceteris paribus* interpretation, formally corresponds to a label ranking function that orders the values of a certain attribute depending on the values of the parents of this attribute (predecessors in the graph representation).

Learning Utility Functions

A natural way to represent preferences is to evaluate the alternatives by means of a utility function. In the object preferences scenario, such a function is a mapping $f : \mathcal{X} \longrightarrow \mathcal{U}$ that assigns a utility degree $f(x)$ to each object x and, thereby, induces a linear order on $\mathcal{X}$; the utility scale $\mathcal{U}$ is usually given by the real numbers $\mathbb{R}$, but sometimes an ordinal scale is preferred (note that an ordinal scale will typically produce many ties, which is undesirable if the target is a ranking). In the label preferences scenario, a utility function $f_i : \mathcal{X} \longrightarrow \mathcal{U}$ is needed for every label λ_i, $i = 1, \ldots, m$. Here, $f_i(x)$ is the utility assigned to alternative λ_i by instance x. To obtain a ranking for x, the alternatives are ordered according to their utility scores, i.e., a ranking $\succ_x$ is derived that satisfies $\lambda_i \succ_x \lambda_j \Rightarrow f_i(x) \geq f_j(x)$.

If the training data offers the utility scores directly, preference learning reduces to a standard regression (up to a monotonic transformation of the utility values) or an ordinal regression problem, depending on the underlying utility scale. This information can rarely be assumed, however. Instead, usually only constraints derived from comparative preference information of the form "This object (or label) should have a higher utility score than that object (or label)" are given. Thus, the challenge for the learner is to find a function that is as much as possible in agreement with a set of such constraints.

For object ranking approaches, this idea has first been formalized by Tesauro (1989) under the name *comparison training*. 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 provide a real-valued evaluation of single states. Similar ideas have also been investigated for training other types of classifiers, in particular support vector machines. We already mentioned Joachims (2002) who analyzed "click-through data" in order to rank documents retrieved by a search engine according to their relevance. Earlier, Herbrich et al. (1998) have proposed an algorithm for training SVMs from pairwise preference relations between objects.

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

Learning Preference Relations

As mentioned before, instead of learning a latent utility function that evaluates individual objects, an alternative approach to preference learning consists of comparing pairs of objects (labels) in terms of a binary preference relation. For object ranking problems, this pairwise approach has been pursued in Cohen et al. (1999). The authors propose to solve object ranking problems by learning a binary preference predicate $Q(x, x')$, which predicts whether x is preferred to x' or vice versa. A final ordering is found in a second phase by deriving a ranking that is maximally consistent with these predictions.

For label ranking problems, the pairwise approach has been introduced in Fürnkranz and Hüllermeier (2003) as a natural extension of *pairwise classification*, a well-known ▶class binarization technique. The idea is to train a separate model (base learner) $\mathcal{M}_{i,j}$ for each pair of labels $(\lambda_i, \lambda_j) \in \mathcal{L}$, $1 \le i < j \le m$; thus, a total number of $m(m-1)/2$ models is needed. For training, a preference information of the form $\lambda_i \succ_x \lambda_j$ is turned into a (classification) example (x, y) for the learner $\mathcal{M}_{a,b}$, where $a = \min(i,j)$ and $b = \max(i,j)$. Moreover, $y = 1$ if $i < j$ and $y = 0$ otherwise. Thus, $\mathcal{M}_{a,b}$ is intended to learn the mapping that outputs 1 if $\lambda_a \succ_x \lambda_b$ and 0 if $\lambda_b \succ_x \lambda_a$:

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

The mapping (1) can be realized by any binary classifier. Instead of a $\{0,1\}$-valued classifier, one can of course also employ a scoring classifier. For example, the output of a probabilistic classifier would be a number in the unit interval $[0,1]$ that can be interpreted as a probability of the preference $\lambda_a \succ_x \lambda_b$.

At classification time, a query $x_0 \in \mathcal{X}$ is submitted to the complete ensemble of binary learners. Thus, a collection of predicted pairwise preference degrees $\mathcal{M}_{i,j}(x)$, $1 \le i,j \le m$, is obtained. The problem, then, is to turn these pairwise preferences into a ranking of the label set $\mathcal{L}$. To this end, different ranking procedures can be used. The simplest approach is to extend the (weighted) voting procedure that is often applied in pairwise classification: For each label λ_i, a score

$$S_i = \sum_{1 \le j \ne i \le m} \mathcal{M}_{i,j}(x_0)$$

is derived (where $\mathcal{M}_{i,j}(x_0) = 1 - \mathcal{M}_{j,i}(x_0)$ for $i > j$), and then all labels are ordered according to these scores. Despite its simplicity, this ranking procedure has several appealing properties. Apart from its computational efficiency, it turned out to be relatively robust in practice and, moreover, it possesses some provable

optimality properties in the case where Spearman's rank correlation is used as an underlying accuracy measure. Roughly speaking, if the binary learners are unbiased probabilistic classifiers, the simple "ranking by weighted voting" procedure yields a label ranking that maximizes the expected Spearman rank correlation (Hüllermeier and Fürnkranz, 2010). Finally, it is worth mentioning that, by changing the ranking procedure, the pairwise approach can also be adjusted to accuracy measures other than Spearman's rank correlation.

Future Directions

As we already mentioned, preference learning is an emerging topic and, as a subfield of machine learning, still in its infancy. In particular, one may expect that, apart from the object and label ranking problems, other settings and frameworks will be studied in the future. But even for object and label ranking as introduced above, there are several open questions and promising lines of future research. The most obvious extension concerns the type of preference structure predicted as an output: For many applications, it is desirable to predict structures which are more general than rankings, e.g., which allow for *incomparability* (partial orders) or *indifference* between alternatives. In a similar vein, the pairwise approach to label ranking has recently been extended to the prediction of so-called "calibrated" rankings in Fürnkranz et al. (2008). A calibrated ranking is a ranking with an additional "zero-point" that separates between a positive and a negative part, thereby integrating the problems of label ranking and multilabel classification.

Cross References

►Classification
►Meta-Learning
►Rank Correlation

Recommended Reading

Aiolli, F., & Sperduti, A. (2010). A preference optimization based unifying framework for supervised learning problems. In J. Fürnkranz & E. Hüllermeier (Eds.), *Preference learning* (pp. 19–40). Springer.

Boutilier, C., Brafman, R., Domshlak, C., Hoos, H., & Poole, D. (2004). CP-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research, 21*, 135–191.

Cohen, W. W., Schapire, R. E., & Singer, Y. (1999). Learning to order things. *Journal of Artificial Intelligence Research, 10*, 243–270.

Dekel, O., Manning, C. D., & Singer, Y. (2004). Log-linear models for label ranking. In S. Thrun, L. K. Saul, & B. Schölkopf, (Eds.), *Advances in neural information processing systems (NIPS-03)* (pp. 497–504). Cambridge: MIT Press.

Doyle, J. (2004). Prospects for preferences. *Computational Intelligence, 20*(2), 111–136.

Fürnkranz, J., & Hüllermeier, E. (2003). Pairwise preference learning and ranking. In N. Lavrač, D. Gamberger, H. Blockeel, & L. Todorovski (Eds.), *Proceedings of the 14th European Conference on Machine Learning (ECML-03), volume 2837 of Lecture Notes in Artificial Intelligence*, Springer, Cavtat, Croatia, pp. 145–156.

Fürnkranz, J., & Hüllermeier, E. (Eds.). (2010). *Preference learning*. Springer.

Fürnkranz, J., & Hüllermeier, E. (2010). Preference learning and ranking by pairwise comparison. In J. Fürnkranz & E. Hüllermeier (Eds.), *Preference Learning* (pp. 63–80). Springer.

Fürnkranz, J., & Hüllermeier, E. (2010). Preference learning: an introduction. In J. Fürnkranz & E. Hüllermeier (Eds.), *Preference Learning* (pp. 1–18). Springer.

Fürnkranz, J., Hüllermeier, E., Loza Mencía, E., & Brinker, K. (2008). Multilabel classification via calibrated label ranking. *Machine Learning, 73*(2), 133–153.

Har-Peled, S., Roth, D., & Zimak, D. (2002). Constraint classification: a new approach to multiclass classification. In N. Cesa-Bianchi, M. Numao, & R. Reischuk, (Eds.), *Proceedings of the 13th International Conference on Algorithmic Learning Theory (ALT-02)* (pp. 365–379), Springer, Lübeck, Germany.

Herbrich, R., Graepel, T., Bollmann-Sdorra, P., & Obermayer, K. (1998). Supervised learning of preference relations. *Proceedings des Fachgruppentreffens Maschinelles Lernen (FGML-98)*, pp. 43–47.

Hüllermeier, E., & Fürnkranz, J. (2010). On predictive accuracy and risk minimization in pairwise label ranking. *Journal of Computer and System Sciences, 76*(1), 49–62.

Hüllermeier, E., Fürnkranz, J., Cheng, W., & Brinker, K. (2008). Label ranking by learning pairwise preferences. *Artificial Intelligence, 172*, 1897–1916.

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

Kamishima, T., Kazawa, H., & Akaho, S. (2010). A survey and empirical comparison of object ranking methods. In J. Fürnkranz & E. Hüllermeier (Eds.), *Preference learning* (pp. 175–195). Springer.

Radlinski, F., Kurup, M., & Joachims, T. (2010). Evaluating search engine relevance with click-based metrics. In J. Fürnkranz & E. Hüllermeier (Eds.), *Preference learning* (pp. 329–353). Springer.

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

Tsochantaridis, I., Hofmann, T., Joachims, T., & Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. *ICML*.