Active Learning of Label Ranking Functions

Klaus Brinker KBRINKER@UNI-PADERBORN.DE
International Graduate School of Dynamic Intelligent Systems, University of Paderborn, 33098 Paderborn, Germany

Abstract

The effort necessary to construct labeled sets of examples in a supervised learning scenario is often disregarded, though in many applications, it is a time-consuming and expensive procedure. While this already constitutes a major issue in classification learning, it becomes an even more serious problem when dealing with the more complex target domain of total orders over a set of alternatives. Considering both the pairwise decomposition and the constraint classification technique to represent label ranking functions, we introduce a novel generalization of pool-based active learning to address this problem.

1. Introduction

The increasing shift from predetermined and static to personalized and highly adaptive systems has affected various areas of application. Techniques to individualize application flows and to incorporate user preferences have produced more efficient systems in the domains of e-commerce (Riecken, 2000), information retrieval and design of user interfaces (Langley, 1997), among others. A fundamental prerequisite of systems which consider individuals rather than predefined standard users is the ability to efficiently acquire accurate preference models.

We consider a special category of preference learning problems, so-called *(label) ranking problems*. The fundamental objective is to learn a mapping from a given input space to the set of total orders over a finite and *a priori* fixed set of *alternatives (labels)*. For example, suppose we are given a set of customers which are represented by features (such as age, income, family status, etc.) and their ordered preferences over a set of car models {Porsche, Toyota, Ford, Lada}. Then, the learning task consists in inducing a mapping from customers to the set of permutations of these car models. We can employ the induced *ranking function* to predict the order of preference for new customers. In contrast to a classification setting, we are not only interested in the top-ranked alternative but in the complete preference order. By incorporating this additional information, we can build more powerful prediction systems which, for instance, are able to make preference suggestions in situations where the top-ranked alternative is currently not available for some reason.

As in the case of multiclass classification, there exist different approaches to reduce ranking problems to binary classification problems. As a straightforward generalization of one-against-one (multiclass) classification, ranking problems can be decomposed into binary classification problems considering all pairwise preferences between two alternatives (Fürnkranz & Hüllermeier, 2003). Each pairwise preference problem is treated independently as a binary classification problem and predictions are made by means of a voting procedure. An alternative approach to the expression of a ranking problem in terms of a single binary classification problem has been proposed by Har-Peled et al. (2002). Transforming the initial ranking problem both involves embedding the training data in a higher dimensional space and expanding single ranking examples into multiple binary classification examples.

Both approaches consider the problem of learning a ranking function in a supervised batch learning scenario. Hence, it is assumed that we are given a training set of examples associated with the corresponding permutations. However, there are many applications in which assigning permutations to examples (herein after referred to as *labeling* in compliance with standard notation) cannot be performed automatically but involves human decisions or costly interviews. Therefore, it is a time-consuming and expensive task. While this already constitutes a major issue in classification learning, it becomes an even more serious problem

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004. Copyright 2004 by the authors.

when dealing with the more complex target domain of the set of permutations: To ask for a customer's top preference is less expensive than to request a complete preference order over all possible alternatives.

The superordinate concept of *active learning* refers to a collection of approaches that aim at reducing this labeling effort (see section 5 for a more detailed discussion). We consider the *pool-based active learning model* [1] (Lewis & Gale, 1994): Starting with only a small amount of labeled examples, the learning algorithm sequentially selects new examples from a finite set of unlabeled examples and requests the corresponding permutations. The crucial point is that by selecting only the most informative examples to be labeled, in many applications it is possible to learn a model by using fewer labeled examples without a significant loss of generalization accuracy in comparison to conventional batch learning based on the entire set of labeled examples.

In the field of kernel machines, active learning has been successfully applied to classification problems to reduce the labeling effort (Tong & Koller, 2000; Campbell et al., 2000; Warmuth et al., 2002). All these approaches are restricted to either binary or multiclass classification problems and do not extend to ranking problems. We propose a novel extension of active learning to ranking problems. Considering both pairwise decomposition (Fürnkranz & Hüllermeier, 2003) and the constraint classification technique (Har-Peled et al., 2002), we propose heuristic strategies to the selection of new training examples. Experimental results indicate a significant reduction of the labeling effort.

This paper is organized as follows: The subsequent section discusses the above stated techniques to express and train ranking functions. In section 3, we investigate active learning in the case of ranking problems and introduce our novel generalization. Section 4 discusses experimental results conducted on a number of synthetic datasets and demonstrates the benefits of our approach. In section 5, we point out references to related research and finally give a conclusion.

2. Ranking Problems

This section recapitulates two techniques to solve ranking problems. Formally, the learning problem that we are investigating can be stated as follows: Based on a given training set of labeled examples

$$T = \{(x_1, y_1), \dots, (x_m, y_m)\} \subset (\mathcal{X} \times \mathcal{S}^{(d)})^m$$

[1]If not noted otherwise, we refer to the *pool-based active learning model* as *active learning* herein after.

with $\mathcal{X}$ denoting a nonempty set and $\mathcal{S}^{(d)}$ being the symmetric group of degree d, i.e.

$$y_i = ([y_i]_1, \dots, [y_i]_d)$$

with $\{[y_i]_1, \dots, [y_i]_d\} = \{1, \dots, d\}$, we seek to induce a function

$$f : \mathcal{X} \to \mathcal{S}^{(d)}$$

to the prediction of new examples. In other words, the objects to be learned are total orders over a finite and *a priori* fixed set of alternatives which are represented as permutations. A permutation y is interpreted as follows: If alternative i precedes alternative j in y, then i is preferred over j. This representation is based on the reasonable assumption that the order over the set of labels is irreflexive and anti-symmetric (for a more detailed discussion see (Fürnkranz & Hüllermeier, 2003)).

Both to increase expressivity and to make use of (typically) highly accurate base classifiers, we embed examples from input space $\mathcal{X}$ using a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. The corresponding kernel feature space is denoted by $\mathcal{F}$ and the feature map by $\phi : \mathcal{X} \to \mathcal{F}$ (Schölkopf & Smola, 2002).

2.1. Constraint Classification

The constraint classification approach (Har-Peled et al., 2002) provides a framework to solve a variety of more complex learning problems, such as ranking and multilabel problems, based on (binary) linear classifiers. We restrict our discussion to ranking problems in the linear case, i.e. $\mathcal{X} = \mathbb{R}^n$ endowed with the canonical dot product $\langle \cdot, \cdot \rangle$ to avoid a lengthy and rather technical presentation. Later on we will comment on how to integrate the concept of kernels.

Let us consider the class of *linear sorting functions*:

$$\begin{aligned} f : \mathbb{R}^n &\to \mathcal{S}^{(d)} \\ x &\mapsto \operatorname*{argsort}_{i=1,\dots,d} \langle w_i, x \rangle \end{aligned}$$

with $w_1, \dots, w_d \in \mathbb{R}^n$ denoting weight vectors of linear functions and argsort returning a permutation of $\{1, \dots, d\}$ where i precedes j if $\langle w_i, x \rangle > \langle w_j, x \rangle$ (in the case of equality, i precedes j if $i < j$).

Transforming the initial ranking problem both involves embedding the training data in a higher dimensional space and expanding single ranking examples into multiple binary classification examples: Let $v(x, i)$ denote an embedding of x in $\mathbb{R}^{nd}$ such that the features of $x = ([x]_1, \dots, [x]_n)$ are copied into the features

$(i-1)n+1, \dots, i\,n$ of the augmented vector and the remaining features are set to 0.

We expand (x, y) into a set $T^+(x, y)$ of $d-1$ positive binary classification examples in $\mathbb{R}^{nd} \times \{-1, +1\}$

$$T^+(x, y) = \bigcup_{i=1,\dots,d-1} \left\{ \left(v(x, [y]_i) - v(x, [y]_{i+1}), +1 \right) \right\}$$

and a set of $d-1$ negative examples

$$T^-(x, y) = \bigcup_{i=1,\dots,d-1} \left\{ \left(v(x, [y]_{i+1}) - v(x, [y]_i), -1 \right) \right\}.$$

The transformed training set T' is defined as the union of all expanded examples:

$$T' = \bigcup_{i=1,\dots,m} T^+(x_i, y_i) \cup T^-(x_i, y_i).$$

Suppose we apply an arbitrary learning algorithm to the training set T' that calculates a separating hyperplane $h(x) = \langle \mathbf{w}, x \rangle$ with $\mathbf{w} \in \mathbb{R}^{nd}$. Furthermore, we consider $\mathbf{w}$ as the concatenation of d n-dimensional vectors $w_1, \dots, w_d$ and by this means define a linear sorting function $f : \mathbb{R}^n \to \mathcal{S}^{(d)}$. Since h is a separating hyperplane, it follows that for all $(t, +1) = \left(v(x, [y]_i) - v(x, [y]_{i+1}), +1 \right) \in T'$,

$$h(t) = \langle \mathbf{w}, t \rangle = \langle w_{[y]_i}, x \rangle - \langle w_{[y]_{i+1}}, x \rangle > 0.$$

Hence, the linear sorting function f correctly arranges the alternatives $[y]_i$ and $[y]_{i+1}$. Since all constraints on consecutive alternatives are encoded as (positive) binary examples, f is consistent with the original ranking training set. In fact, to ensure consistency in this case, we do not need the expansion into negative examples. While the expansion into both positive and negative examples makes this framework applicable regardless of the underlying training algorithm, we can exploit the fact that the training set is symmetric around the origin. In the case of support vector machines, the one-class algorithm proposed by Schölkopf et al. (2001) can be modified in a straightforward fashion to work on the positive set only. However, we do not make use of this modification because our reasoning is based on a $C-$parametrization whereas (Schölkopf et al., 2001) use a $\nu-$parametrization.

Apart from theoretical analysis, one should not implement the constraint classification framework by explicitly expanding examples. It is more efficient to store constraints imposed by consecutive alternatives and references to original examples in expanded examples and make use of a suitable (meta-)kernel. Furthermore, the standard kernelization technique can be incorporated at this place.

2.2. Pairwise Ranking

Pairwise ranking (Fürnkranz & Hüllermeier, 2003) is a generalization of one-against-one (multiclass) classification which learns a separate binary classifier for each of the $d(d-1)/2$ pairs of alternatives. Each binary classifier h_{ij} (with $1 \le i < j \le d$) decides for a given example whether alternative i or j is preferred. The training set for h_{ij} consists of the complete set of feature vectors $x_1, \dots, x_m$ with the class label y of each example (x, y) being assigned depending on whether i precedes j in y or vice versa.

To predict a new ranking, we determine the classifications of all binary classifiers $h_{ij}(x) \in \{-1, +1\}$ and interpret the outcome as a vote for alternative i or j. Finally, all possible alternatives are sorted in descending order with respect to the sum of votes. In the case of ties, though it might be suboptimal, we prioritize alternatives with smaller indices. Formally, this strategy can be stated as follows:

$$x \mapsto \operatorname*{argsort}_{i=1,\dots,d} \sum_{j=i+1}^{d} \max(h_{ij}(x), 0) + \sum_{j=1}^{i-1} \max(-h_{ji}(x), 0).$$

Pairwise ranking provides a framework that is applicable without any further assumptions on the underlying binary classifier. We consider support vector machines as base classifiers. Whenever there is a close decision between two alternatives, i.e. the given example is close to the classification boundary, it is not necessarily a clever strategy to assign a complete vote to one of them. Therefore, in addition to the above stated *standard voting strategy*, we investigate a modified strategy: Using an approach proposed by Platt (1999), we estimate posterior (positive) class probabilities $h'_{ij}(x) \in [0, 1]$ instead of class labels and assign partial votes to both alternatives:

$$x \mapsto \operatorname*{argsort}_{i=1,\dots,d} \sum_{j=i+1}^{d} h'_{ij}(x) + \sum_{j=1}^{i-1} (1 - h'_{ji}(x)).$$

We refer to this method as *probabilistic voting*.

3. Active Learning

This section introduces novel heuristic active learning criteria for both the constraint classification and the pairwise decomposition technique.

We derive a selection criterion for the constraint classification method based on the *version space model* for binary classification problems: Let us consider a linearly separable (in feature space) binary classification problem. Note that we can deal with noisy, linearly nonseparable data in an elegant way by adding some constant $\nu > 0$ to the diagonal elements of the kernel matrix, $k(x_i, x_j) + \delta_{ij}\nu$, such that the training set becomes linearly separable when using the L2-loss (Shawe-Taylor & Cristianini, 1999).

The nonempty set

$$\mathcal{V} \overset{\text{def}}{=} \{w \in \mathcal{F} \,|\, \text{sign}(\langle w, \phi(x_i)\rangle) = y_i \\ \text{for } i = 1, \ldots, m \quad \text{and} \quad \|w\| = 1\}$$

which consists of all (normalized) weight vectors corresponding to linear classifiers in feature space which separate the training set without errors is called *version space* (Mitchell, 1982). We can view learning as a search problem within version space: Each training example (x_i, y_i) limits the volume of the version space because to correspond to a consistent classifier a weight vector has to satisfy

$$\text{sign}(\langle w, \phi(x_i)\rangle) = y_i \quad \Leftrightarrow \quad y_i \langle w, \phi(x_i)\rangle > 0.$$

In other words, consistent solutions are restricted to a halfspace whose boundary is the hyperplane with normal vector $y_i\phi(x_i)$. For a fixed feature vector $\phi(x_i)$, the class label y_i determines the orientation of the halfspace. Moreover, $\mathcal{V}$ is the intersection of m halfspaces (a convex polyhedral cone) with the unit sphere in feature space $\mathcal{F}$.

A classical result from the theory of convex sets states that any halfspace containing the center of mass of a convex set comprises at least $1/e$ of the overall volume (Grünbaum, 1960). Assume we are able to repeatedly select unlabeled examples which correspond to restricting hyperplanes passing exactly through the current center of mass of version space w_{center}. Then, independent of the actual class label, the volume of version space is reduced exponentially in terms of the number of labeled examples.[2]

To derive a practical selection criterion, we have to make a number of approximations: It is computationally expensive to calculate the center of mass in high dimensional spaces. Therefore, the center of mass is approximated by the center of the largest radius ball $\tilde{w}_{\text{center}}$ in version space. When working on a normalized set of examples, this approximation corresponds to a support vector machine. Moreover, since we are only given a finite set of unlabeled examples to choose from, mostly we will not be able to find an example which exactly meets the above stated criterion. Hence, we select that unlabeled example whose restricting hyperplane is closest to the approximation of the center of mass, i.e. examples minimizing $|\langle \tilde{w}_{\text{center}}, \phi(x)\rangle|$.

[2]More precisely, for this argument to hold, we have to consider an augmented (convex) version space as the intersection with the unit ball.

Assume that the requested example is labeled such that the larger part of the current version space remains. The ratio of volume reduction still approaches at least $1 - 1/e$ the closer the restricting hyperplane to the center of mass. For a convex set in isotropic position, any halfspace at distance t from the center of mass contains at least $\frac{1}{e} - t$ of its volume (Bertsimas & Vempala, 2002). Based on analogous reasoning, the margin $y\langle \tilde{w}_{\text{center}}, \phi(x)\rangle$ of a labeled example (x, y) can be considered as an approximate measure of the reduction of volume where lower values correspond to a higher reduction and negative values correspond to the case where the smaller part of the version space remains. Considering a best worst-case approach, we obtain minimization of $\max_{y\in\{-1,+1\}} y\langle \tilde{w}_{\text{center}}, \phi(x)\rangle$ as selection criterion which is a reformulation of the former criterion in the binary case. Apart from the version space model which has been considered in (Tong & Koller, 2000), there are additional theoretical justifications for this approach (Campbell et al., 2000).

Coming back to the constraint classification framework, the notion of the margin can be generalized in a straightforward fashion as the minimum margin within the set of expanded binary examples (Har-Peled et al., 2002):

Definition (Generalized Margin). *The margin $\delta : \mathcal{X} \times \mathcal{S}^{(d)} \to \mathbb{R}$ of a ranking example (x, y) with respect to the linear sorting function f is defined as*

$$\delta(x, y) = \min_{i=1,\ldots,d-1} \left\langle w_{[y]_i}, \phi(x)\right\rangle - \left\langle w_{[y]_{i+1}}, \phi(x)\right\rangle.$$

If a support vector machine is used as the component learner on the expanded binary training set to solve the corresponding ranking problem, it maximizes this generalized margin. Moreover, this definition reduces to the standard margin for ranking problems with $d = 2$ (which can be considered as binary classification problems).

It is straightforward to derive an upper bound on the margin of a ranking example (x, y):

$$\begin{aligned} \delta(x, y) &\leq \max_{y'\in\mathcal{S}^{(d)}} \min_{i=1,\ldots,d-1} \left\langle w_{[y']_i}, \phi(x)\right\rangle - \left\langle w_{[y']_{i+1}}, \phi(x)\right\rangle \qquad (1) \\ &= \min_{\substack{i,j=1,\ldots,d \\ i\neq j}} |\langle w_i, \phi(x)\rangle - \langle w_j, \phi(x)\rangle| \\ &\overset{\text{def}}{=} \delta^+(x). \end{aligned}$$

Note that this bound is tight in the sense that for every x there exists a y such that equality holds in (1): If and only if $y = \text{argsort}_{i=1,\ldots,d} \langle w_i, x\rangle$, then $\delta(x, y) = \delta^+(x)$. Therefore, given an unlabeled example x, $\delta^+(x)$ evaluates to the worst-case margin for all choices of $y \in \mathcal{S}^{(d)}$.

Now, in a straightforward fashion, we can generalize selection of new training examples based on minimum worst-case margin from classification learning to the problem of learning ranking functions: For all unlabeled examples, we evaluate $\delta^+(x)$ and request the correct ranking corresponding to that example with minimum worst-case margin. From a different point of view, we select examples yielding (approximately) maximum worst-case volume reduction of version space.

The pairwise ranking technique conducts an expansion into binary classification problems considering all pairs of alternatives. Hence, we are given *a set* of independently treated binary problems which is not directly amenable to analysis in the binary version space model. In order to derive a heuristic selection criterion, we consider a best worst-case approach with respect to the minimum binary margin on the set of binary examples corresponding to a given ranking example. In contrast to the constraint classification framework, margins have to be evaluated based on *different* binary classifiers. More precisely, for a *labeled* ranking example, we have to consider the real-valued output of $d(d-1)/2$ binary classifiers h_{ij} (with $1 \leq i < j \leq d$) to calculate the minimum binary margin. However, for an *unlabeled* example, it is computationally infeasible to consider all $d!$ possible permutations to evaluate the minimum margin on the set of binary examples in the worst-case. Therefore, we approximate the permutation yielding minimum margin by the predicted permutation of an unlabeled ranking example. In the case of the probabilistic voting strategy, we consider an analogous best worst-case approach with respect to minimum binary class probabilities. To summarize the selection criterion for the pairwise ranking model, we calculate the predicted rankings for all unlabeled examples and select that unlabeled example which achieves minimum margin (class probability) on the set of binary problems.

4. Experiments

4.1. Experimental Setting

To evaluate the efficiency of our novel selection criteria, we conducted a number of experiments using support vector machines (Chang & Lin, 2001) as binary linear learners. Due to the lack of suitable real-world datasets, we generated artificial data considering three different settings.

Linear: We replicate a setting proposed by Fürnkranz and Hüllermeier (2003) from the field of expected utility theory: An expected utility maximizing agent is given a set $\{1, \ldots, d\}$ of alternative actions to choose from. The agent faces a problem of *decision under uncertainty* with alternative i yielding a utility $[U]_{ij} \in \mathbb{R}$ if the world is in state $\omega_j \in \Omega = \{\omega_1, \ldots, \omega_n\}$. The probability of state ω_j is denoted by $[p]_j$ and, therefore, the expected utility of alternative i is given by $E(i) = \sum_{j=1}^{n} [p]_j \, [U]_{ij}$. Thus, giving rise to a natural order over the set of alternative actions. We assume the set of alternatives to be in decreasing order with respect to expected utility in the following. Let us assume the probability vector $p = ([p]_1, \ldots, [p]_n)$ to be the feature vector of a ranking example while the number of alternatives d and the set of world states Ω being fixed and the $d \times n$ utility matrix U having independently uniformly distributed entries $[U]_{ij} \in [0, 1]$. Then, for a given probability vector p the above stated decision-theoretic scenario gives rise to an order over the set of alternative actions. Now, a set of m feature vectors is independently drawn from a uniform distribution over $\{p \in \mathbb{R}^n \mid p \geq 0, \, [p]_1 + \cdots + [p]_n = 1\}$ and assigned to corresponding permutations to generate a ranking dataset. Note that this setting corresponds to a noise-free scenario in the constraint classification framework since for a given feature vector p an alternative way to express the corresponding ranking is $y = \operatorname{argsort}_{i=1,\ldots,d} \langle u_i, p \rangle$ (with u_i denoting the i-th row vector of U). We conducted our experiments on this dataset using a linear kernel with penalty parameter $C = 100$.

MinMax: This setting considers a modified (non-linear) preference relation generated by $E(i) = \min_{j=1,\ldots,n} \max([U]_{ij}, 1 - [p]_j)$. It can be viewed as a special case of a pessimistic criterion to evaluate the worth of an alternative in a possibilistic decision framework (Dubois et al., 2001). To stay consistent with the herein stated assumptions, for each feature vector p a single feature is randomly selected and set to 1. On this problem, we used an RBF-kernel with $\gamma = 0.1$ and penalty parameter $C = 100$.

QRank: We train a Naive Bayes classifier on the VEHICLE multiclass dataset from the UCI repository. For each example the set of possible class labels (alternatives) is ordered with respect to the *a posteriori* probabilities assigned by the Naive Bayes classifier. From a more abstract point of view, we consider the problem of learning a qualitative replication of the order over a set of alternatives induced by a probabilistic classifier. As for the former setting, we use an RBF-kernel with the default value of $\gamma = 1/\#\text{features}$ and $C = 100$.

For both the Linear and the MinMax scenario, we fixed the number of input features to $n = 10$. For each number of alternatives $d \in \{5, 10, 15, 20\}$, we generated 100 different datasets consisting of 2000 examples, each dataset originating from a different utility matrix U. Each dataset was randomly split into a training set and a test set of equal size. In the QRank scenario, we cannot sample new data for each run. Instead, the same underlying dataset was randomly split into a training set and a test set of equal size for each run in compliance with comparable research on real-world data. While new training examples were selected from the training sets, the generalization accuracy was estimated on the test sets.

The well-known Spearman rank correlation coefficient (rank correlation) was used as the evaluation metric on the true rankings y and predicted rankings y':

$$c(y, y') = 1 - \frac{6 \sum_{i=1}^{d} ([y]_i - [y']_i)^2}{d(d^2 - 1)}.$$

The rank correlation evaluates to -1 for reversed preference orders and to $+1$ for identical orders. Moreover, the rank correlation was averaged over all examples in a test set. Due to space restrictions, we do not comment on alternative evaluation measures.

In addition to our novel active learning generalization of the pairwise and constraint classification technique, we investigated random selection of new examples as a baseline strategy for each of the approaches. We started with a randomly selected set of 10 labeled examples in all experiments and sequentially selected 90 examples using the different selection criteria. The rank correlation was evaluated after every 10 rounds and finally the results were averaged over all 100 datasets generated in each of the settings.

4.2. Experimental Results

Remember that the choice of kernel is not optimized with respect to the different techniques. Therefore, we compare random learning with their active counterparts separately for each approach and do not focus on quantitative comparison of different techniques. Table 1 shows average rank correlations and corresponding standard errors of the mean for different numbers of labeled examples. Due to space restrictions, for the Linear and MinMax scenario, detailed results are stated only for the number of alternatives being fixed to $d = 10$.

Linear: All active strategies consistently outperform their random counterparts for all choices of the number of alternatives. While there is a substantial increase of accuracy in the case of $d = 5$ alternatives, it becomes marginal in the case of constraint classification with the number of alternatives increasing. In contrast to this, for the pairwise decomposition techniques, the absolute rise in accuracy at fixed numbers of labeled examples increases with the number of alternatives.

MinMax: The active standard pairwise strategy achieves a level of accuracy for 20 labeled examples (independent of d) that is very close to that of random learning of 100 examples. For the probabilistic pairwise technique there is a substantial increase of accuracy for $d = 5, 10$, whereas for $d = 15, 20$ the active strategy is superior at the beginning while random learning slightly outperforms its active counterpart at the end. In the case of constraint classification, we observed a pattern similar to in the former setting: Active learning consistently outperforms random learning. The gain of accuracy decreases with the number of alternatives increasing.

QRank: Again, active constraint classification learning consistently outperforms random learning. Compared to the former approach, both pairwise decomposition strategies yield an even more significant decrease of the labeling effort: The reduction is roughly two-fold, e.g. actively learning circa 50 examples yields the same estimated generalization accuracy than randomly learning 100 examples.

As in classification learning, the efficiency of active learning clearly depends on the given ranking problem. In our experiments, active learning based on the constraint classification approach consistently outperforms random learning. Both active learning based on standard and probabilistic pairwise decomposition yields a more significant relative reduction of the labeling effort in most of the experiments. Furthermore, the computational effort to solve a ranking problem based on the constraint classification technique was more than two orders of magnitude higher than that of both pairwise decomposition techniques. Therefore, our experiments suggest that it is favorable to make use of the pairwise decomposition approach when actively learning a ranking function.

5. Related Work

Beyond the (label) ranking model, there are alternative preference models which consider different learning scenarios. In the field of statistical decision theory and mathematical economics, the problem of learning a (preference) utility function (von Neumann & Morgenstern, 1944) of a given individual is referred to as preference elicitation. While in this case the objective is to assign real-valued utility scores to examples according to a single user's preferences, ranking functions assign a finite preference order to each example. More generally, in the former model, examples corre-

Table 1. Experimental results

SETTING	STRATEGY*	10	20	30	40	50	60	70	80	90	100
Linear ($d = 10$)	RANDOM CC	0.419 ±0.010	0.594 ±0.007	0.697 ±0.005	0.753 ±0.004	0.793 ±0,004	0.821 ±0.003	0.842 ±0.003	0.856 ±0.003	0.869 ±0.002	0.879 ±0.002
	ACTIVE CC	0.412 ±0.010	0.602 ±0.007	0.710 ±0.005	0.770 ±0.004	0.809 ±0.004	0.837 ±0.003	0.855 ±0.003	0.872 ±0.002	0.885 ±0.002	0.896 ±0.002
	RANDOM PW	0.356 ±0.013	0.463 ±0.011	0.532 ±0.010	0.582 ±0.009	0.622 ±0.008	0.651 ±0.007	0.677 ±0.007	0.699 ±0.006	0.717 ±0.006	0.730 ±0.005
	ACTIVE PW	0.361 ±0.012	0.476 ±0.011	0.566 ±0.008	0.627 ±0.007	0.671 ±0.007	0.705 ±0.006	0.731 ±0.005	0.753 ±0.005	0.769 ±0.005	0.784 ±0.005
	RANDOM PPW	0.394 ±0.013	0.513 ±0.010	0.586 ±0.009	0.640 ±0.007	0.676 ±0.006	0.703 ±0.005	0.723 ±0.005	0.741 ±0.005	0.758 ±0.004	0.771 ±0.004
	ACTIVE PPW	0.389 ±0.012	0.544 ±0.009	0.644 ±0.007	0.705 ±0.006	0.744 ±0.005	0.771 ±0.004	0.791 ±0.004	0.807 ±0.004	0.819 ±0.004	0.831 ±0.003
MinMax ($d = 10$)	RANDOM CC	0.502 ±0.007	0.634 ±0.007	0.718 ±0.006	0.767 ±0.005	0.803 ±0.005	0.833 ±0.004	0.853 ±0.004	0.868 ±0.003	0.881 ±0.003	0.891 ±0.003
	ACTIVE CC	0.506 ±0.008	0.683 ±0.007	0.763 ±0.005	0.812 ±0.005	0.844 ±0.005	0.868 ±0.004	0.887 ±0.004	0.900 ±0.004	0.910 ±0.004	0.919 ±0.004
	RANDOM PW	0.614 ±0.011	0.807 ±0.009	0.886 ±0.007	0.917 ±0.006	0.928 ±0.005	0.933 ±0.004	0.936 ±0.004	0.937 ±0.004	0.938 ±0.004	0.939 ±0.004
	ACTIVE PW	0.621 ±0.011	0.931 ±0.004	0.936 ±0.004	0.939 ±0.003	0.940 ±0.003	0.942 ±0.003	0.944 ±0.003	0.945 ±0.003	0.945 ±0.003	0.946 ±0.003
	RANDOM PPW	0.480 ±0.010	0.720 ±0.008	0.828 ±0.007	0.878 ±0.006	0.901 ±0.005	0.912 ±0.005	0.920 ±0.004	0.923 ±0.004	0.924 ±0.004	0.926 ±0.004
	ACTIVE PPW	0.511 ±0.010	0.752 ±0.008	0.874 ±0.005	0.907 ±0.004	0.918 ±0.004	0.922 ±0.004	0.926 ±0.004	0.928 ±0.004	0.930 ±0.004	0.932 ±0.004
QRank	RANDOM CC	0.613 ±0,008	0.677 ±0,004	0.718 ±0,004	0.741 ±0,003	0.759 ±0,003	0.773 ±0,003	0.782 ±0,003	0.793 ±0,003	0.800 ±0,002	0.807 ±0,003
	ACTIVE CC	0.599 ±0.007	0.687 ±0.004	0.741 ±0.003	0.764 ±0.003	0.782 ±0.002	0.798 ±0.002	0.810 ±0.002	0.819 ±0.002	0.828 ±0.002	0.837 ±0.002
	RANDOM PW	0.655 ±0.008	0.710 ±0.005	0.743 ±0.004	0.762 ±0.003	0.779 ±0.003	0.789 ±0.003	0.799 ±0.002	0.807 ±0.002	0.817 ±0.002	0.823 ±0.002
	ACTIVE PW	0.649 ±0.007	0.737 ±0.004	0.785 ±0.003	0.812 ±0.002	0.823 ±0.002	0.831 ±0.002	0.838 ±0.002	0.843 ±0.002	0.847 ±0.001	0.849 ±0.001
	RANDOM PPW	0.600 ±0.008	0.673 ±0.005	0.708 ±0.003	0.726 ±0.003	0.741 ±0.003	0.756 ±0.003	0.766 ±0.003	0.775 ±0.003	0.785 ±0.002	0.792 ±0.002
	ACTIVE PPW	0.615 ±0.009	0.703 ±0.004	0.754 ±0.003	0.785 ±0.002	0.804 ±0.002	0.816 ±0.002	0.828 ±0.002	0.832 ±0.002	0.836 ±0.002	0.841 ±0.002

* CONSTRAINT CLASSIFICATION (CC), STANDARD PAIRWISE (PW), PROBABILISTIC PAIRWISE (PPW).

spond to decision alternatives, whereas in the latter model a finite set of alternatives is *a priori* fixed and examples correspond to different individuals. Learning a utility function can be considered as a regression problem. In the field of kernel methods, Crammer and Singer (2002) introduced an online algorithm to learn a utility function on an ordinal scale which is used to order a set of new examples with respect to their ordinal utility scores. This problem is also referred to as ordinal regression and has been investigated in the batch setting by Herbrich et al. (2000). Beyond the class of regression-based preference models, Cohen et al. (1999) propose a two-stage approach to preference learning: In stage one, they learn a probabilistic preference function on pairs of given examples which in stage two is used to order a given set of new examples in some (approximately) optimal sense.

In the field of active learning, there are two principle categories of approaches: So-called *query learning* (Angluin, 1988) refers to a learning model where the learning algorithm is given the ability to generate new examples and request the corresponding true class labels. Whereas in *selective sampling* the learner is restricted to request labels of a finite set of examples (*pool-based model*) or the learning algorithm has to decide whether to request the corresponding true labels for sequentially presented single examples (*stream-based model*). The well-studied Bayesian *query-by-committee* (Seung et al., 1992; Freund et al., 1997) approach considers the latter scenario and decides to request class labels based on the disagreement within a set of Gibbs classifiers.

6. Conclusion

We introduced novel extensions of pool-based active learning to label ranking problems based on both the constraint classification technique and pairwise decomposition of preferences. Experimental results clearly indicate that active learning can significantly reduce the labeling effort in ranking learning. Considering that is more expensive to label ranking rather than classification examples, the benefit of *active* ranking learning becomes even more evident.

Acknowledgments

We would like to thank Eyke Hüllermeier and Chih-Len Lin for helpful suggestions and comments.

References

Angluin, D. (1988). Queries and concept learning. *Journal of Machine Learning*, *2*, 319–342.

Bertsimas, D., & Vempala, S. (2002). Solving convex programs by random walks. *Proceedings of the 34th ACM Symposium on the Theory of Computing (STOC '02)* (pp. 109–115). Montreal.

Campbell, C., Cristianini, N., & Smola, A. (2000). Query learning with large margin classifiers. *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)* (pp. 111–118).

Chang, C.-C., & Lin, C.-J. (2001). *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

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

Crammer, K., & Singer, Y. (2002). Pranking with ranking. *Advances in Neural Information Processing Systems 14* (pp. 641–647). Cambridge, MA: MIT Press.

Dubois, D., Prade, H., & Sabbadin, R. (2001). Decision-theoretic foundation of qualitative possibility theory. *European Journal of Operational Research*, 459–459.

Freund, Y., Seung, H. S., Shamir, E., & Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine Learning*, *28*, 133–168.

Fürnkranz, J., & Hüllermeier, E. (2003). Pairwise preference learning and ranking. *Proceedings of the 14th European Conference on Machine Learning* (pp. 145–156). Cavtat, Croatia: Springer-Verlag.

Grünbaum, B. (1960). Partitions of mass-distributions and convex bodies by hyperplanes. *Pacific J. Math.*, *10*, 1257–1261.

Har-Peled, S., Roth, D., & Zimak, D. (2002). Constraint classification: A new approach to multiclass classification and ranking. *Advances in Neural Information Processing Systems 15 (NIPS)*.

Herbrich, R., Graepel, T., & Obermayer, K. (2000). *Advances in large margin classifiers*, chapter Large margin rank boundaries for ordinal regression, 115–132. Cambridge, MA: MIT Press.

Langley, P. (1997). Machine learning for adaptive user interfaces. *Proceedings of the 21st German Annual Conference on Artificial Intelligence* (pp. 53–62).

Lewis, D. D., & Gale, W. A. (1994). A sequential algorithm for training text classifiers. *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval* (pp. 3–12). Dublin, IE: Springer Verlag.

Mitchell, T. M. (1982). Generalization as search. *Journal of Artificial Intelligence*, *18*, 203–226.

Platt, J. (1999). Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. *Advances in Large Margin Classifiers* (pp. 61–74). Cambridge, MA: MIT Press.

Riecken, D. (2000). Personalized views of personalization. *Communications of the ACM*, *43*, 26–28.

Schölkopf, B., Platt, J., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, *13*, 1443–1472.

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels: Support vector machines, regularization, optimization, and beyond.* Cambridge, MA: MIT Press.

Seung, H. S., Opper, M., & Sompolinsky, H. (1992). Query by committee. *Proceedings of the Fifth Anual ACM Workshop on Computaional Learning Theory* (pp. 287–294).

Shawe-Taylor, J., & Cristianini, N. (1999). Further results on the margin distribution. *Proceedings of the twelfth annual conference on Computational learning theory* (pp. 278–285). Santa Cruz, California, United States: ACM Press.

Tong, S., & Koller, D. (2000). Support vector machine active learning with applications to text classification. *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 999–1006). Morgan Kaufmann, San Francisco, CA.

von Neumann, J., & Morgenstern, O. (1944). *Theory of games and economic behavior*. Princeton University Press.

Warmuth, M. K., Rätsch, G., Mathieson, M., Liao, J., & Lemmen, C. (2002). Active learning in the drug discovery process. *Advances in Neural information processing systems* (pp. 1449–1456).