Learning User Preferences for Sets of Objects

Marie desJardins MARIEDJ@CS.UMBC.EDU
Eric Eaton EEATON1@CS.UMBC.EDU
University of Maryland Baltimore County, Computer Science and Electrical Engineering Department, 1000 Hilltop Circle, Baltimore, MD 21250 USA

Kiri L. Wagstaff KIRI.WAGSTAFF@JPL.NASA.GOV
Machine Learning and Instrument Autonomy Group, Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109 USA

Abstract

Most work on preference learning has focused on pairwise preferences or rankings over individual items. In this paper, we present a method for learning preferences over *sets* of items. Our learning method takes as input a collection of positive examples—that is, one or more sets that have been identified by a user as desirable. Kernel density estimation is used to estimate the value function for individual items, and the desired set diversity is estimated from the average set diversity observed in the collection. Since this is a new learning problem, we introduce a new evaluation methodology and evaluate the learning method on two data collections: synthetic blocks-world data and a new real-world music data collection that we have gathered.

1. Introduction

Many interactions between humans and computers involve a search for information or items. For example, a World Wide Web search engine can produce a list of webpages that are ranked by their relevance to a specified search query. The underlying assumption is that the user is searching for a specific piece of information, and that the pages can be ranked in terms of their likelihood of containing the desired information.

In contrast, there are many applications in which the user instead wishes to obtain an optimal *set* of items. Examples include building a music playlist or selecting a sports team from this year's draft picks. The obvious approach of ranking all items and then picking the top k does not, as it turns out, always yield the optimal subset. Items in a set can interact in ways that increase (via complementarity), or decrease (via redundancy or incompatibility), the overall valuation[1] of a set. For example, while a user may have a favorite music artist, he or she may not want a party music playlist composed solely of that artist's work. Likewise, some of the top k players available may play the same position, so they cannot all be selected together. This phenomenon has been referred to as "the portfolio effect" (Burke, 2002) or "dependent relevance" (Zhai et al., 2003).

Recently, we developed a language, DD-PREF, that allows the specification of set-based preferences. We also proposed a greedy algorithm that accounts for the portfolio effect while searching for the most satisfying set (desJardins & Wagstaff, 2005). One problem with such a language is that it may be difficult for users to explicitly specify their preferences quantitatively. The primary contribution of the current paper is a method for *learning* set-based user preferences from example sets that implicitly express those preferences. Learning these preferences will enable existing subset-selection methods to automatically identify or generate additional satisfying subsets, such as novel music playlists. While several methods have been proposed for learning to rank individual items (see Section 2), to our knowledge no methods have yet been devised for learning set-based preferences. This problem is particularly challenging, because it involves learning from positive examples only, often with only a few examples (possibly just one). We also present a new data set (classical music playlists from radio station websites) and the experimental methodology we devised for comparing different preference learning approaches.

[1] "Value" refers to a specific feature value, while "valuation" refers to how highly preferred a set or value is.

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006.

2. Related Work

The main areas of related research involve preference specification, satisfaction, and learning.

Preference Specification. Most work in this area has focused on preferences over individual items, expressed as judgments about the relative valuation of two items (Herbrich et al., 1998; Cohen et al., 1999; Freund et al., 2003). Brafman et al. (2005) allow users to specify conditional preferences in terms of a minimum or desired number of objects in the set that have particular feature values. Our DD-PREF language permits users to specify set-based preferences on a per-feature basis (desJardins & Wagstaff, 2005).

There has been a significant amount of work in the multi-agent systems community on representing preferences over sets of items in combinatorial auctions and on efficient methods for clearing such auctions (Boutilier, 2002; Cramton et al., 2006). However, all of this work is essentially propositional: that is, it assumes that individual items are atomic, not represented by a set of attributes as in our research. As a result, preferences cannot be generalized (learned) or computed over new sets of items.

Preference Satisfaction. Barberà et al. (2004) refer to the problem of identifying the best subset of size k as "fixed-cardinality ranking." They discuss the problem of computing a ranking over sets, given only pairwise judgments between items. In general, finding the optimal subset is NP-complete (Cohen et al., 1999; Price & Messinger, 2005), so the most common approach is to use a greedy heuristic that incrementally adds items to the subset (Cohen et al., 1999; Zhai et al., 2003; desJardins & Wagstaff, 2005; Price & Messinger, 2005).

The *portfolio effect* occurs when the valuation of a set is not equal to the sum of its component item valuations. This can be modeled in various ways, such as trading off "relevance" against "novelty" for the subtopic retrieval problem (Zhai et al., 2003) or by measuring the "marginal relevance" of each new item to be added to a set of results (Carbonell & Goldstein, 1998). DD-PREF accounts for the portfolio effect by allowing the user to explicitly specify the relative importance of "diversity" and "depth".

Price and Messinger (2005) tackle a slightly different problem, in which the user is assumed to be interested in one specific item, and the goal is to return the subset of items most likely to contain the desired item. In contrast, our focus is on situations in which the user wishes to obtain a specific kind of subset, where all of the component items (and their interactions) are important.

Preference Learning. To our knowledge, no methods currently exist for learning user preferences over sets. Methods for learning whether one individual item is preferred to another include Hedge, an online ensemble approach (Cohen et al., 1999); RankBoost (Freund et al., 2003); RankNet (Burges et al., 2005); and probabilistic modeling (Chu & Ghahramani, 2005). RankProp (Caruana et al., 1996) and PRank (Crammer & Singer, 2001) instead use regression to map individual items to target valuations for direct ranking.

3. Preference Learning

In this section, we review the DD-PREF language and how to assess the valuation of a given set (that is, how well the set satisfies the specified preference). Next, we describe our major contribution, which is a method for automatically learning these set-based preferences, given one or more example sets provided by the user.

3.1. Representing Preferences in DD-PREF

Preferences in DD-PREF are represented as tuples: $P = \langle \vec{\mathbf{q}}, \vec{\mathbf{d}}, \vec{\mathbf{w}}, \alpha \rangle$. For feature f with values in the set V_f, $q_f : V_f \rightarrow [0,1]$ is the desired *depth* (preferred feature values); $d_f \in [0,1]$ is the desired *diversity* (preferred distribution of values across the desired range); and $w_f \in [0,1]$ is the feature preference weight. The $\alpha \in [0,1]$ parameter specifies the relative importance of diversity versus depth, across all features.

Depth. The desired depth is specified by a quality function q_f for each feature f, which specifies the valuation of each possible feature value for f. For example, Figure 1 shows food preferences for three different users, as a simple illustrative example. The first user, who prefers healthy food, has a quality function with a bimodal distribution for protein: some desired items are high in protein, and some are low. This individual also prefers foods that are low in fat. In contrast, the "Atkins" user prefers high-protein and low-carb foods, and the "junk food" user prefers low-protein and high-carb (sugar) foods. The depth valuation of a set s is defined as a weighted sum of the average valuations for each feature:

$$\mathcal{V}_{dep}(s|P) = \sum_f \left(P.w_f \ \frac{1}{|s|} \sum_{x \in s} P.q_f(x^f) \right), \qquad (1)$$

where x^f is the value for feature f of item x. Highly weighted features are underlined in Figure 1.

Diversity. The desired diversity for each feature, d_f, is a number between 0 and 1 that specifies how

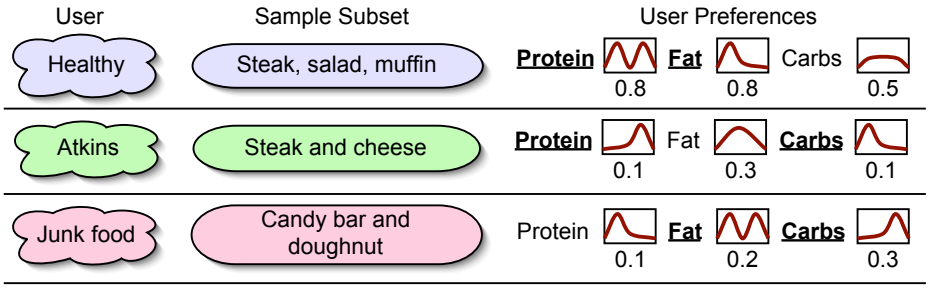


Figure 1. Three examples of user preferences when selecting foods for a meal. Each user's abstract preference (left) is specified quantitatively (right). The quality functions (q_f) for each feature (protein, fat, and carbohydrates) are shown as small graphs, with the preferred diversities (d_f) below each one. Highly weighted features are underlined.

evenly distributed the values should be. In Figure 1, the "Atkins" user desires very little diversity in the amount of protein or carbs present, while the "healthy" user wants high diversity in the amount of protein present: the steak is high in protein, but the salad and muffin are low. In this case (for a feature with high diversity and a multimodal distribution), the preference will be best satisfied by a collection of foods at different peaks in the quality function. For a feature with a multimodal distribution but *low* diversity, such as the "junk food" user's fat preference, the preference would be best satisfied by a collection of foods at any *single* peak in the quality function. The candy bar and doughnut are high in fat, but this user would be equally pleased with jelly beans and soda, which have none.

The diversity valuation of a set s is a weighted sum of how closely the observed diversity, $\text{div}_f(s)$, matches the desired diversity, $P.\vec{\mathbf{d}}$:

$$\mathcal{V}_{div}(s|P) = \sum_f P.w_f(1 - (P.d_f - \text{div}_f(s))^2). \quad (2)$$

The observed diversity $\text{div}_f(s)$ is a measure of the "spread" of a set's values for feature f. We use the definition introduced by desJardins and Wagstaff (2005), but other definitions are possible.

We define the valuation of a set s with respect to preference P as a trade-off between diversity and depth, controlled by the diversity weight parameter, α:

$$\mathcal{V}_P(s) = (1 - P.\alpha)\,\mathcal{V}_{dep}(s|P) + P.\alpha\,\mathcal{V}_{div}(s|P) \quad (3)$$

3.2. Learning DD-PREF Preferences

This work is motivated by the observation is that it is often much easier for a user to specify examples of their preferences, rather than trying to convert an abstract concept, such as "healthy" food, into quantitative preferences.

Problem definition: Given a data collection S consisting of one or more training sets s_i, learn a preference Q that approximates the (implicit) preference P that was used to generate S.

To learn preference Q for the data collection $S = \{s_1, \ldots, s_n\}$, we first estimate the desired depth and diversity for each feature, and then we estimate the optimal feature weights. In this work, we do not address learning the diversity parameter, α. In some applications, an appropriate α may already be known. In the experiments in this paper, we simply set $\alpha = 0.5$, evenly weighting diversity and depth.

Learning Depth Preferences, $\vec{\mathbf{q}}$. We treat the problem of learning the quality functions as a probability density estimation problem. Specifically, we assume that the frequency with which a feature value appears in the training data is proportional to its valuation by the user. As with any machine learning approach, this represents an implicit assumption that the set of features used to represent the domain is appropriate and sufficient.

We estimate the quality functions q_f using kernel density estimation (KDE) (Duda et al., 2001), a well established method for estimating probability density functions. KDE models each observed value with a Gaussian distribution, then sums the Gaussians generated by all data items to produce a single, smoothed estimate of the distribution. For this work, we used the kernel density estimator described by the Analytical Methods Committee (2001). We obtain q_f by normalizing the estimated density function P_{KDE} by the likelihood of the most likely value:

$$q_f(v) = \frac{P_{KDE}(v)}{\max_{v'} P_{KDE}(v')}. \quad (4)$$

This estimator automatically selects a value for the width σ of the Gaussian distributions.

Learning Diversity Preferences, $\vec{\mathbf{d}}$. We use the maximum *a posteriori* (MAP) estimate of the desired diversity d_f by calculating the average observed diversity for feature f over the sets in collection S: $d_f = \frac{1}{|S|} \sum_{s_i \in S} \mathrm{div}_f(s_i)$.

Learning Feature Weights, $\vec{\mathbf{w}}$. As a final step, we (locally) optimize the feature weights $\vec{\mathbf{w}}$ using BFGS bounded optimization (Gill et al., 1981; Gill & Murray, 1976), provided in the Weka machine learning toolkit (Witten & Frank, 2005). BFGS is a quasi-Newton algorithm for locally minimizing an objective function; the bounded-optimization form of the algorithm provided with Weka adds bounds constraints on the variables. Since the actual valuations $\mathcal{V}_P$ of the training sets s_i are unknown, we set their target valuations to 1.0, a perfect score. We then use BFGS to find the weights $\vec{\mathbf{w}}$ that minimize the objective function:

$$\sum_{s_i \in S} (\mathcal{V}_P(s_i) - \mathcal{V}_Q(s_i))^2 = \sum_{s_i \in S} (1 - \mathcal{V}_Q(s_i))^2.$$

4. Data Sets and Metrics

4.1. Data Sets

Artificial Blocks. To test our ability to learn user preferences in a simple domain, we used a blocks data set in which each item is represented by four features: size (a real value from 0 to 100), color (represented as integers from 0–6), number of sides (an integer value from 3 to 20), and bin (representing sequential locations in a storage area; an integer from 0 to 100).

Classical Music. A realistic domain in which subset selection tasks constantly arise is that of music playlist creation. DJs at radio stations, clubs, and parties must put together playlists that capture the relevant preferences of the audience, radio show theme, or party attendees. We collected a data set that consists of 2352 songs from classical music radio stations. Each song is represented by 28 features:

Feature	Type/Units
Composition date	Year
Composer birthdate	Year
Duration	Minutes
Tempo	Beats per minute
Bark spectrum	Loudness at 24 frequencies (Zwicker & Fastl, 1990)

The dates and durations were obtained by looking up each song in the online "allmusic" database,[2] and the tempo and Bark spectrum values were obtained using BpmDJ.[3]

[2] `http://www.allmusic.com/`

4.2. Preferences

Block Preferences. We tested four different preferences for our experiments. In contrast to the simple preferences we previously evaluated (desJardins & Wagstaff, 2005), in which q_f could only be specified as a range of permissible values, $[min, max]$, we also experimented with more complicated value functions with linear or bimodal distributions. A value of 1.0 for q_f means that all values in V_f are equally valuable.

Castle: We want blocks to build a castle. We need large blocks for structure and small blocks for decoration. We want blocks of similar colors, though we do not care which color is chosen, and the blocks should have few sides. Location does not matter. Thus, we have the following feature preferences $P_f = \langle q_f, d_f, w_f \rangle$:

$$\begin{array}{lcccc} P_{size} & = & \langle \max\left(\frac{100-x}{100}, \frac{x}{100}\right)^2, & 1.0, & 1.00\rangle \\ P_{color} & = & \langle 1.0, & 0.2, & 0.50\rangle \\ P_{sides} & = & \langle [3,8], & 1.0, & 0.75\rangle \end{array}$$

Child: We are choosing blocks for a child. We want a variety of multi-colored medium-sized blocks for grasping, with few sides, located fairly close together.

$$\begin{array}{lcccc} P_{size} & = & \langle [10,50], & 1.0, & 1.0\rangle \\ P_{color} & = & \langle 1.0, & 1.0, & 0.8\rangle \\ P_{sides} & = & \langle [3,6], & 1.0, & 0.8\rangle \\ P_{bin} & = & \langle 1.0, & 0.2, & 0.4\rangle \end{array}$$

Mosaic: We want to create a mosaic with the blocks. We want a variety of blocks of various shapes, with an emphasis on small simple blocks. We want similar, but non-identical colors, and the location of the blocks is not important. The values of size and numsides decrease linearly from 1 to 0 across the range of values. Size is the most important, then color, and finally number of sides.

$$\begin{array}{lcccc} P_{size} & = & \langle \frac{100-x}{100}, & 0.80, & 1.0\rangle \\ P_{color} & = & \langle 1.0, & 0.75, & 0.8\rangle \\ P_{sides} & = & \langle \frac{17-(x-3)}{17}, & 1.00, & 0.6\rangle \end{array}$$

Tower: We want to build a uniform tower. We want large similar-sided blocks of uniform color with a limited number of sides; the location of the blocks is not important.

$$\begin{array}{lcccc} P_{size} & = & \langle [50,100], & 0.1, & 1.0\rangle \\ P_{color} & = & \langle 1.0, & 0.0, & 1.0\rangle \\ P_{sides} & = & \langle [4,8], & 0.0, & 1.0\rangle \end{array}$$

[3] `http://bpmdj.sourceforge.net/`

Music Preferences. To test our ability to learn preferences from example subsets, we collected radio station playlists from two time periods: 1400 songs were collected during an 8-week period in Fall 2005, and another 952 songs were collected during a three-week period prior to Christmas 2005. This latter collection represents a distinctly different preference from the other playlists, since it contains primarily holiday music. We used playlists from three radio stations:

- `http://www.allclassical.org/`, All Classical (KBPS)
- `http://minnesota.publicradio.org/`, Minnesota Public Radio (MPR)
- `http://www.cpr.org/`, Colorado Public Radio (CPR)

Each day's playlist consists of 17 to 53 songs (depending on song duration and the amount of on-air time that day).

4.3. Evaluation Metrics and Methodology

For this new problem of learning preferences over sets, the metrics and methodology must enable evaluation of two claims: first, that a learned preference Q closely approximates the true preference P, when P is known (akin to recall); and second, that the learned preference can be used to distinguish high-valued from low-valued subsets (akin to precision).

Retrieval Similarity. We use two measures of learning performance to compare P and Q. First, we identify s_P, the best subset of the test data according to P, and s_Q, the best subset according to Q. We then compute $\mathcal{V}_P(s_P)$ and $\mathcal{V}_P(s_Q)$ to determine how highly valued the two selected sets are, according to the true preference P.

Functional Similarity. Second, to determine how well P and Q agree with each other, we measure their functional similarity as follows. Let $\vec{\mathbf{v_P}}$ and $\vec{\mathbf{v_Q}}$ be the vector of values assigned by P and Q, respectively, to each s in a collection of subsets S. We then compute the correlation $r(\vec{\mathbf{v_P}}, \vec{\mathbf{v_Q}})$. For a sufficiently large collection S, the computed correlation provides a good estimate of preference similarity.

Preference Precision. To assess the precision of Q, or how accurately it assigns low values to sets that are not preferred, we propose the use of perturbation studies. In such a study, a highly valued subset s_P is iteratively perturbed by replacing a member of the set with some other item (either randomly selected, or drawn from some source other than P). If Q is selective, then we would expect the values assigned by Q to these perturbed sets to drop as more items are replaced.

Methodology. For each trial, we divided the available data into f disjoint collections (folds) of a uniform size. Given true preference P, we used the wrapper-greedy method (desJardins & Wagstaff, 2005) to extract the best subset of items s_P from each fold, yielding f example subsets for P. We used a variation on leave-one-out cross-validation by training preference Q on f' subsets ($f' \in [1, f-1]$) and testing on the held-out subset. We report the average performance across all held-out subsets.

For each of the blocks world experiments, we created a data set using uniform sampling in the feature space. The four preferences we used were described in Section 4.2. For the music experiments, the folds were composed of songs instead of blocks. Rather than manually defining preference P, we instead inferred it, for each trial, from a real, randomly selected playlist.

For each experiment, the number of items in the underlying data set (n), number of items in the training and test sets (k), number of trials (division into folds), number of folds (f), and fold size are shown in Table 1.

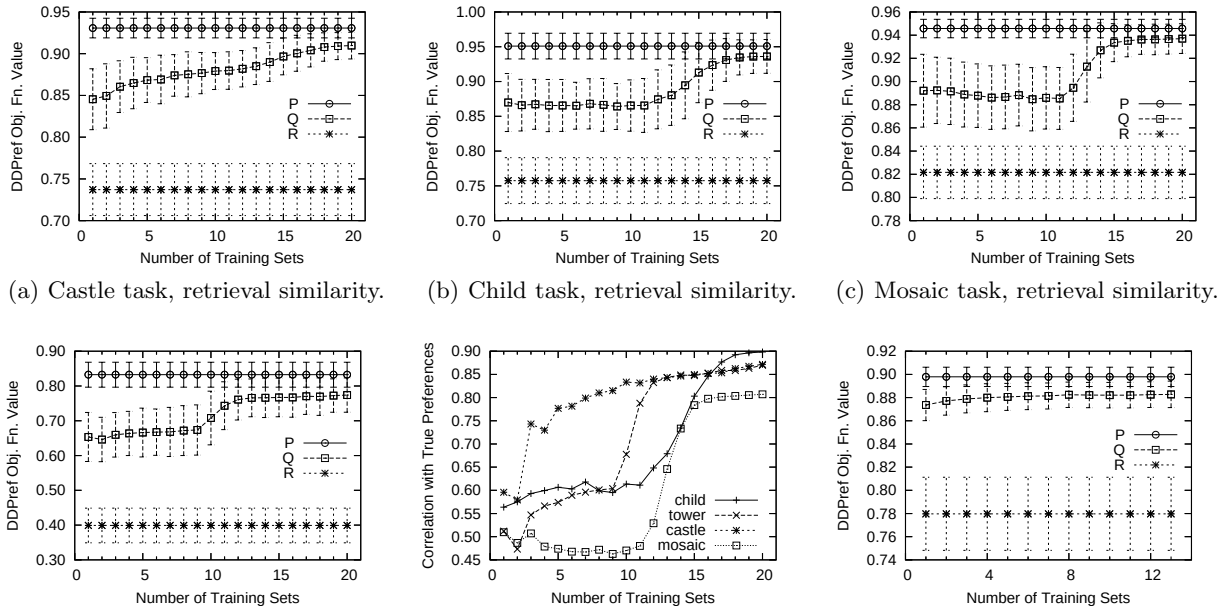
Table 1. Summary of experimental parameters.

Exp.		n	k	# trials	f	fold size
1	blocks	2100	10	20	21	100
1	music	1400	5	20	14	100
2	blocks	1000	10	100	10	100
2	music	2352	5	100	1	2352

5. Experimental Results

In this section, we present the results of two experiments that were designed to evaluate our ability to infer and characterize user preferences over subsets. Experiment 1 (Section 5.1) evaluates learning performance for the blocks and music domains in terms of objective function values and functional similarity between true and learned preferences. Experiment 2 (Section 5.2) shows the results of perturbation experiments in both domains.

Although we do not give timing results, the complexity of the preference learning algorithm is roughly linear in the number of training sets. The main variability of the algorithm occurs in the weight estimations, when minimizing the sum of squared errors over the training sets. The quasi-Newton BFGS method has very rapid convergence properties, and we have not noticed a significant slowdown as the number of training sets was increased.



(a) Castle task, retrieval similarity. (b) Child task, retrieval similarity. (c) Mosaic task, retrieval similarity.

(d) Tower task, retrieval similarity. (e) Block tasks, functional similarity over 21 random subsets. (f) Music playlists, retrieval similarity.

Figure 2. Experiment 1: Learning curves evaluating the similarity between true (P) and learned (Q) preferences.

5.1. Experiment 1: How Accurate are the Learned Preferences?

In this experiment, we examine the question of whether we can accurately infer user preferences. In Figures 2(a)–2(d), we plot the retrieval similarity for P and Q as a function of training collection size, for the four blocks world tasks. We also show the valuation R that P assigns to a random subset as a lower baseline. (Note that the retrieval similarity of P and R do not vary with the number of training sets.) The error bars are placed at one standard deviation from the mean. For all four tasks, as more training data is used, the learned preferences Q more closely approximate the valuations obtained by P.

Figure 2(e) shows the functional similarity (correlation) of the true and learned preferences on a collection of 21 random subsets, as a function of training collection size. The correlations generally increase with more training data. They show much more dramatic (though also more variable) improvement as the size of the training collection increases, compared to the objective function value increases shown in previous figures. The correlation plot, since it is evaluated over a random selection of sets, shows early learning of the proper valuation of both high-quality and low-quality subsets.

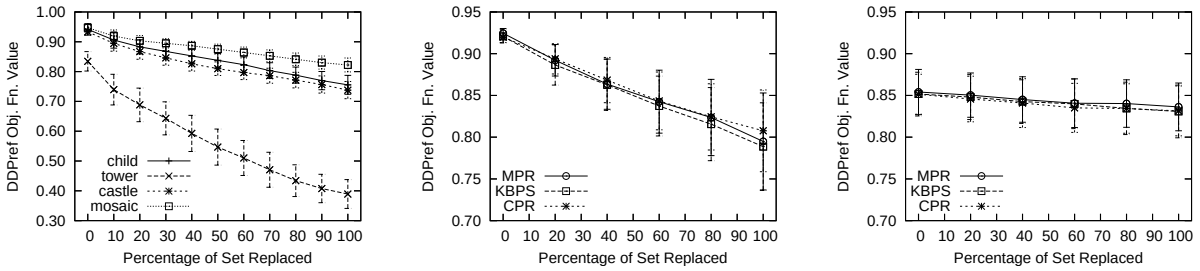
We conducted a similar experiment with the non-holiday music playlists. For this experiment, since we do not have access to the DJ's true preferences, we chose one of the real playlists randomly, of length 5, and learned P from it. We then ran the same experiment as in the blocks world domain.

The learning curve for the music experiment is shown in Figure 2(f). Learned preference Q gives much better results than random; however, more training data only increases performance very slightly. One possible explanation is that the underlying song data is very homogeneous, so that the playlists are all very similar, and additional playlists do not add much new information. Experiment 2, in the next section, gives some evidence that this may in fact be the case.

5.2. Experiment 2: How Selective are the Learned Preferences?

Experiment 1 showed that we are able to learn preferences Q that closely match the behavior of the original preferences P. However, we would also like to know how selective those preferences are. That is, given a highly valued subset, how much does the valuation degrade if we slightly perturb it?

The learning parameters for Experiment 2 are shown in Table 1. We used P to find the best subset s_P for each fold, then replaced members of this subset with blocks from the data set that were not originally in s_P. Figure 3(a) shows $\mathcal{V}_P$ as a function of the percentage of blocks replaced. As expected, the quality drops as



(a) Block tasks. (b) Music domain, preferences learned from a single playlist. (c) Music domain, preferences learned from ten playlists.

Figure 3. Experiment 2: Preference precision evaluated on increasingly perturbed subsets.

selected blocks are replaced with random blocks.

We performed a similar experiment with the music data. Figure 3(b) shows the results. For each curve, a preference Q was learned from a single playlist from the specified radio station, then the playlist was perturbed by replacing one song at a time with a random song. For these experiments, we used non-holiday playlists to learn the preferences, but used both holiday and non-holiday music for replacement. Each curve is averaged over 100 trials. As in the blocks world perturbation experiment, the valuation of the set drops as more of the songs are replaced, though not as dramatically.

In contrast, Figure 3(c) shows the results when each station's preference function was learned from ten playlists. Again, the graph shows valuations for playlists from that station as the songs are replaced with random songs. Here, we see very little drop in valuation as the number of replacements increases. We can infer that although a single playlist may reflect a unique distribution, the aggregate playlists from each of the three stations are very similar. In fact, when the replacement process uses only non-holiday music, the valuations do not drop at all. (These results were omitted due to space limitations.) To further explore this issue, we are currently gathering more specialized and varied music collections (Mozart festival music from late January 2006 and rock music from other sources), to determine whether these contain more readily distinguishable preferences.

The observation that the radio stations' playlists are fairly homogeneous is supported by a correlation analysis. Table 2(a) shows the functional similarity (correlation) of preferences learned from a single playlist for the three different radio stations. The correlation across stations is seen to be moderate, averaging around 0.55. In Table 2(b), however, we show the functional similarity of preferences learned from 10 playlists from each radio station. These are seen to be very highly correlated, averaging around 0.87.

6. Conclusions and Future Work

We have presented a method for learning set-based preferences and shown that it is effective at learning a variety of preferences in synthetic and real-world domains. We are currently extending the music data set so that we can examine preferences across more heterogeneous collections. In particular, we plan to evaluate preference learning on three specialized data sets: playlists during the 2005 Christmas holiday season, playlists during Mozart's birth week in 2006, and popular rock music. Other future work includes developing a method for estimating the diversity parameter α and incorporating background knowledge about the nature of the quality functions.

Acknowledgments

This work was partly supported by NSF grant #0325329 and was partly carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Thanks to James MacGlashan and Ryan Carr for gathering the music data set.

References

Analytical Methods Committee (2001). *Representing data distributions with kernel density estimates* (Technical Report AMC Technical Brief No. 4). Royal Society of Chemistry.

Barberà, S., Bossert, W., & Pattanaik, P. K. (2004). Ranking sets of objects. In S. Barberà, P. J. Hammond and C. Seidl (Eds.), *Handbook of utility theory*, vol. 2: Extensions, chapter 17. Springer.

Boutilier, C. (2002). Solving concisely expressed com-

Table 2. Functional similarity (correlation) between preferences learned from different radio stations' playlists.

(a) One training set.

	MPR	KBPS	CPR	Random
MPR	1.0000	0.5535	0.5896	0.5565
KBPS		1.0000	0.5527	0.5097
CPR			1.0000	0.5726
Random				1.0000

(b) Ten training sets.

	MPR	KBPS	CPR	Random
MPR	1.0000	0.8739	0.8907	0.8625
KBPS		1.0000	0.8719	0.8524
CPR			1.0000	0.8740
Random				1.0000

binatorial auction problems. *Proceedings of the Eighteenth National Conference on Artificial Intelligence* (pp. 359–366).

Brafman, R. I., Domshlak, C., Shimony, S. E., & Silver, Y. (2005). Preferences over sets. *Working Notes of the IJCAI-05 Workshop on Advances in Preference Handling*.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to rank using gradient descent. *Proceedings of the Twenty-Second International Conference on Machine Learning* (pp. 89–96).

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, *12*, 331–370.

Carbonell, J., & Goldstein, J. (1998). The use of MMR, diversity-based reranking for reodering documents and producing summaries. *Proceedings of the 21st Annual ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)* (pp. 335–336). ACM Press.

Caruana, R., Baluja, S., & Mitchell, T. (1996). Using the future to 'sort out' the present: Rankprop and multitask learning for medical risk evaluation. *Advances in Neural Information Processing Systems (Proceedings of NIPS 95)* (pp. 959–965). MIT Press.

Chu, W., & Ghahramani, Z. (2005). Preference learning with Gaussian processes. *Proceedings of the Twenty-Second International Conference on Machine Learning* (pp. 137–144).

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. (2001). Pranking with ranking. *Proceedings of the Neural Information Processing Systems Conference* (pp. 641–647).

Cramton, P., Shoham, Y., & Steinberg, R. (Eds.). (2006). *Combinatorial auctions*. MIT Press.

desJardins, M., & Wagstaff, K. L. (2005). DD-PREF: A language for expressing preferences over sets. *Proceedings of the Twentieth National Conference on Artificial Intelligence* (pp. 620–626).

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification*. Wiley-Interscience. Second edition.

Freund, Y., Iyer, R., Schapire, R. E., & Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, *4*, 933–969.

Gill, P. E., & Murray, W. (1976). *Minimization subject to bounds on the variables*. National Physical Laboratory.

Gill, P. E., Murray, W., & Wright, M. H. (1981). *Practical optimization*. Academic Press.

Herbrich, R., Graepel, T., Bollmann-Sdorra, P., & Obermayer, K. (1998). Learning preference relations for information retrieval. *Proceedings of the Learning for Text Categorization AAAI Workshop* (pp. 83–86).

Price, B., & Messinger, P. R. (2005). Optimal recommendation sets: Covering uncertainty over user preferences. *Proceedings of the Twentieth National Conference on Artificial Intelligence* (pp. 541–548).

Witten, I. H., & Frank, E. (2005). *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann. Second edition.

Zhai, C., Cohen, W. W., & Lafferty, J. (2003). Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval. *Proceedings of SIGIR'03* (pp. 10–17).

Zwicker, E., & Fastl, H. (1990). *Psychoacoustics, facts and models*. Springer-Verlag.