Automated Acquisition of User Preferences

L. Karl Branting
Department of Computer Science
University of Wyoming
Box 3682
Laramie, Wy 82071
(307) 766-4258
karl@index.uwyo.edu

Patrick S. Broos
Department of Astronomy
Pennsylvania State University
0525 Davey Laboratory
University Park, PA 16802
(814) 863-5505
patb@astro.psu.edu

Abstract

Decision support systems often require knowledge of users' preferences. However, preferences may vary among individual users or be difficult for users to articulate. This paper describes how user preferences can be acquired in the form of *preference predicates* by a learning apprentice system and proposes two new instance-based algorithms for preference predicate acquisition: *1ARC* and *Compositional Instance-Based Learning* (CIBL). An empirical evaluation using simulated preference behavior indicated that the instance-based approaches are preferable to decision-tree induction and perceptrons as the learning component of a learning apprentice system if representation of the relevant characteristics of problem-solving states requires a large number of attributes, if attributes interact in a complex fashion, or if there are very few training instances. Conversely, decision-tree induction or perceptron learning is preferable if there are a

small number of attributes and the attributes do not interact in a complex fashion unless there are very few training instances. When tested as the learning component of a learning apprentice system used by astronomers for scheduling astronomical observations, both CIBL and decision-tree induction rapidly achieved useful levels of accuracy in predicting the astronomers' preferences.

1 Introduction

A central impediment to the construction of knowledge-based systems is the high cost of knowledge base development and maintenance. One approach to reducing these costs is to design systems that can acquire knowledge by observing human problem-solving steps during normal use of the system. Systems that engage in this form of learning are termed *learning apprentice systems* [Mitchell et al., 1985]. Learning apprentice systems have been developed for VLSI design [Mahadevan et al., 1993], acquisition of "interface agents" [Maes and Kozierok, 1993], and calendar management [Dent et al., 1992].

An important form of knowledge that can be acquired by observing users' decisions is knowledge of users' preferences. In configuration tasks such as design or scheduling, for example, there may be numerous configurations that satisfy all applicable hard constraints. Users may nevertheless strongly prefer some configurations to others. For example, in document layout there are typically countless arrangements in which all textual and graphical elements fit within the dimensions of the page and no two elements occupy the same space. However, these arrangements may differ significantly

in balance, contrast, emphasis, and other factors of importance to editors and graphic designers. Choosing among layouts requires a model of the relative desirability of layouts as a function of these factors. Similarly, in the domain of scheduling ground-based telescope observations there are typically many different schedules that satisfy all hard constraints (such as not pointing the telescope at the sun or below the horizon, not scheduling two observations at the same time, *etc.*). However, such schedules may differ significantly in factors such as the airmass,[1] research priority of each scheduled observation, or total telescope slew time. Choosing among such schedules requires a model of the relative desirability of schedules as a function of their relevant attributes.

Automated acquisition of users' preferences is particularly important when users differ in their individual preferences or are unable to articulate the precise preference criteria that they use. Under these circumstances the most promising approach is to develop a learning apprentice system capable of forming "personalized knowledge-based systems" [Dent et al., 1992].

The approach to learning-apprentice acquisition of user preferences described in this paper is appropriate for a variety of tasks—typified by design and configuration problems—in which (1) users can identify the relevant characteristics of problem-solving states, (2) these state characteristics can be adequately represented as an attribute vector, but (3) users differ as to or are unable to articulate evaluation criteria for problem solving states in terms of these attributes.

[1] The airmass of an observation is a measure of the amount of atmosphere between the star and the observer. Airmass can be minimized by observing a star at the time midway between its rising time and setting time.

The next section describes previous approaches to the problem of acquiring preference criteria and proposes two novel algorithms for this task: *1ARC* and *Compositional Instance-Based Learning* (CIBL). Section three describes a series of learning experiments that identify the factors that control the relative performance of 1ARC and CIBL in comparison to decision-tree induction and perceptron learning. Section four describes a prototype application of preference learning in an advising system for astronomical scheduling, explains why acquisition of user preferences is important for an automated assistant for this task, and presents preliminary results indicating that automated preference acquisition is feasible for tasks of this type.

2 Techniques for Acquiring User Preferences

Knowledge of users' preferences can be expressed as a *preference predicate* [Utgoff and Saxena, 1987] $P_Q(x, y) \equiv [Q(x) > Q(y)] \equiv$ "state x is preferred to state y", where $Q(s)$ is an evaluation function that expresses the "quality" of state s. Information about P_Q can be acquired by a learning apprentice in the form of pairs (x, y) such that $P_Q(x, y)$. For example, each time a learning apprentice suggests a state s_1 and the user rejects s_1 in favor of some other state s_2, the apprentice has an opportunity to acquire the training instance $P_Q(s_2, s_1)$. A learning apprentice can therefore acquire a user's criteria for the relative desirability of alternative states by learning a preference predicate P_Q from a set of training instances $P_Q(s_i, s_j)$ produced by the user during normal use of the system.

Previous approaches to acquisition of preference predicates from sets of training

instances have used inductive learning methods to form generalizations from sets of training instances [Utgoff and Saxena, 1987, Utgoff and Clouse, 1991]. One approach has been to use decision tree induction algorithms, such as ID3 [Quinlan, 1986], to induce a general representation for P_Q. An alternative approach, termed the *state preference method*, uses parameter adjustment to learn a set of feature weights $\mathbf{W}$ such that for every training instance, $P_Q(x, y)$, $\mathbf{W}(\mathbf{F}(x) - \mathbf{F}(y)) > 0$, where $\mathbf{F}(n)$ is a vector of numeric attributes representing state n [Utgoff and Clouse, 1991]. The state-preference method can be implemented through perceptron learning.

However, these approaches are not well-suited to all possible preference predicates. The state preference method presupposes that the underlying evaluation function Q has an accurate linear approximation. However, in many domains preference predicates have no linear approximation [Callan et al., 1991]. Decision tree induction algorithms such as ID3 are suitable for nonlinearly separable data. However, the performance of decision tree induction algorithms has been shown to be sometimes weaker than that of instance-based algorithms when the training set is sparse or the concept being learned is "irregular" [Aha, 1992]. Under these circumstances, instanced-based learning methods are sometimes more effective.

2.1 Instance-Based Learning of Preference Predicates

Instance-based learning (IBL) is a strategy in which concepts are represented by exemplars rather than by generalizations induced from those exemplars [Aha, 1990, Stanfill and Waltz, 1986]. Perhaps the simplest form of instance-based learning is

k-nearest-neighbor (k-NN) classification, which classifies a new instance according to the majority classification of its k nearest neighbors in feature space. In most recent IBL systems, $k = 1$ [Aha, 1990].

1ARC is a 1-NN strategy for learning preference predicates that uses a representation of training instances as arcs in feature space. For example, on a two dimensional feature space $S = \Re^2$, instances $\{P_Q(A,B), P_Q(C,D), P_Q(E,F)\}$ can be represented as shown in Figure 1 by arcs $\overleftarrow{AB}$, $\overleftarrow{CD}$, and $\overleftarrow{EF}$ (where $\overleftarrow{XY} \equiv P_Q(X,Y)$).

Ranking a new pair of objects, X and Y, is equivalent to determining whether $P_Q(X,Y)$ or $P_Q(Y,X)$ is satisfied. The 1ARC algorithm begins by finding the training instance that best matches the hypothesis $P_Q(X,Y) \equiv \overleftarrow{XY}$. The dissimilarity between $\overleftarrow{XY}$ and a training instance is measured by the sum of the Euclidean distances between (1) Y and the tail of the training arc and (2) X and the head of the training arc. The dissimilarity between $\overleftarrow{XY}$ and the training arc that it matches most closely, *i.e.*, for which the dissimilarity is least, is a measure in the confidence in the hypothesis $P_Q(X,Y)$. In Figure 1, for example, the training arc $\overleftarrow{EF}$ best matches $\overleftarrow{XY}$ with a dissimilarity of $dist(Y,F) + dist(X,E)$ represented by the dotted lines.

In the same way, 1ARC then finds the best match and confidence measure for the alternate hypothesis $P_Q(Y,X)$. The hypothesis with the strongest measure of confidence determines 1ARC's estimate of the ranking between X and Y. In this case, $\overleftarrow{XY}$ matches training arc $\overleftarrow{EF}$ more strongly than $\overleftarrow{YX}$ matches any training arc, so 1ARC concludes that $P_Q(X,Y)$.

A limitation of k-NN algorithms, such as 1ARC, when applied to the predicate

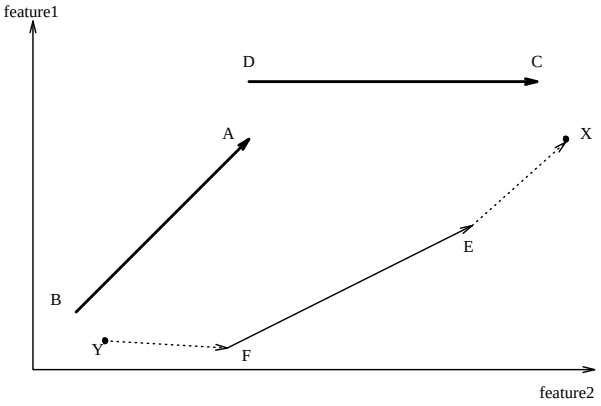

Figure 1: The best match to $\overleftarrow{XY}$ found by 1ARC.

learning task is that they are unable to exploit the transitivity of preference predicates. For example, given the situation in Figure 1, it should be possible to conclude $P_Q(X, Y)$ by the reasoning "X is close to C; C is preferred to D; D is close to A; A is preferred to B; B is close to Y". However, the majority vote policy of standard k-NN methods does not permit reasoning involving the serial composition of multiple instances.

2.2 Compositional Instance-Based Learning

CIBL (Compositional Instance-Based Learning) is an extension of 1ARC that permits multiple training instances to be composed to rank a new pair of objects. Like 1ARC, CIBL ranks two new objects, X and Y, by determining whether it has greater confidence in the path from X to Y or in the path from Y to X. CIBL differs from 1ARC in that it can construct a path between two new objects by sequentially connecting multiple training instances. CIBL uses the Dijkstra algorithm [Aho et al., 1974] to

find the least-cost path, assuming that the path from the tail to the head of a training instance has zero cost and all other portions of the path have a cost equal to their Euclidean length. Such a path seeks to follow a contour of the underlying evaluation function having positive slope.

For example, given the situation shown in Figure 2, CIBL begins by searching for the minimum cost path from Y to X, supporting the hypothesis $P_Q(X, Y)$. The cost of this path is the sum of the Euclidean lengths of the "gaps" G1, G2, and G3. In a similar fashion, a path is constructed from X to Y. The path with lower cost (in this case, the path from Y to X) determines the best estimate of the ranking of X and Y.

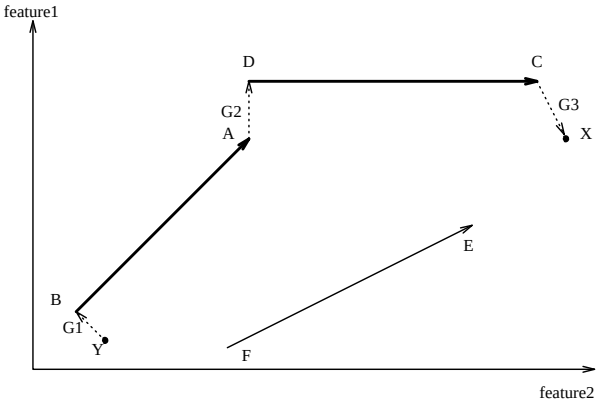

Figure 2: The best match to hypothesis $\overleftarrow{XY}$ found by CIBL.

2.3 Characteristics of Arc Representation of Preference Instances

The arc representation of preference instances used by 1ARC and CIBL differs in several important respects from the representations formed by the state-preference

method and from decision trees. First, a single arc is sufficient to summarize an entire linear region of feature space, *i.e.,* the 1ARC procedure using a single arc parallel to the gradient of a linear function will correctly rank any testing instance in terms of that linear function.[2] By contrast, the standard method for applying decision trees to numerical feature spaces, [Quinlan, 1986], (described in more detail below) entails division of feature space into hyper-rectangular regions. For example, given a single instance $P_Q(X, Y)$, this approach finds a hyperplane that divides feature space into two regions: one containing X and one containing Y. A testing instance in which each point is in a different region may be correctly ranked. However, if both points are in the same region, then the rank of the points will always be the same regardless of the relative position of the points. Thus, the ranking of such testing instances will be no better than chance. As the number of training instances goes up, the number of regions into which feature space is divided by decision-tree induction increases, and the likelihood that both points in a testing instance are in the same region therefore decreases. However, one would expect that decision-tree induction would have low accuracy given small numbers of training instances.

A second characteristic of the arc representation of preference instances is that it is not limited to linear quality functions, since it can include multiple arcs. By contrast, the state-preference method is poorly suited to nonlinear quality functions.

Third, the arc representation has the advantage that it permits the inherent transitivity of preference instances to be exploited through the CIBL approach. A typical

[2] A proof of this assertion is set forth in Appendix A.

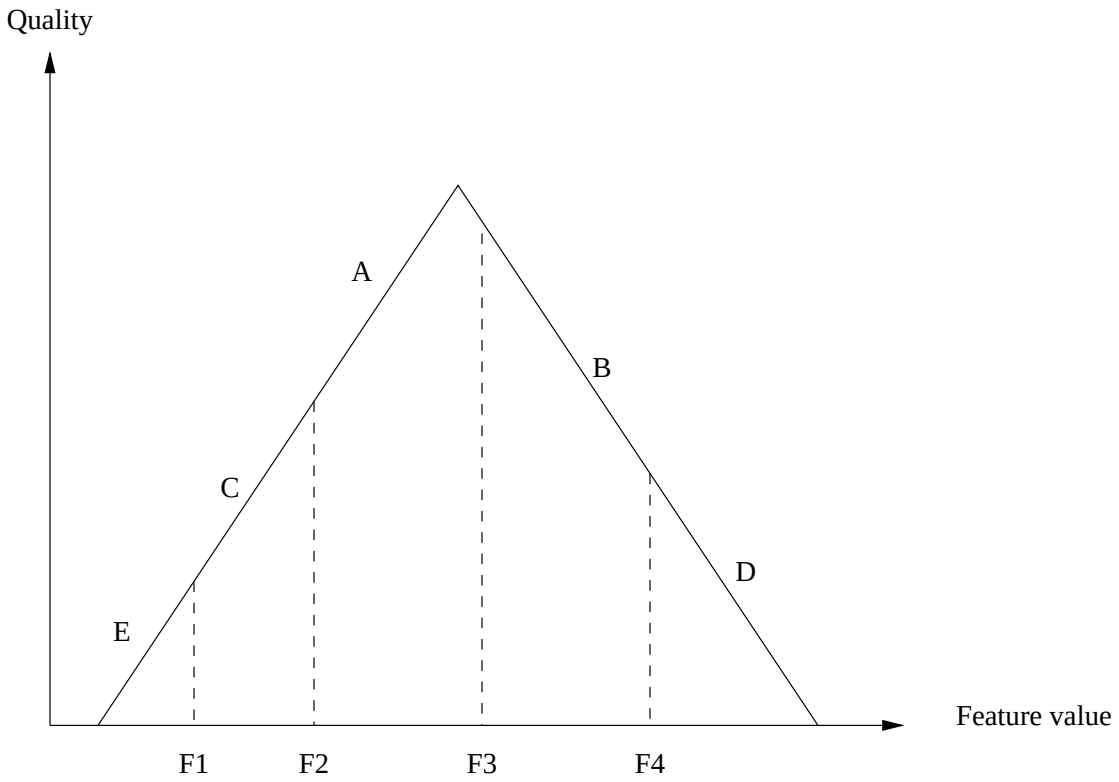

Figure 3: A quality function Q of one feature and instances A,B,C,D, and E.

situation in which use of transitivity can increase accuracy is depicted in Figure 3, which shows a quality function, Q, of one variable. The vertical axis is quality, and the horizontal axis is the feature value. Q exhibits a change in the sign of its derivative in the middle of the range depicted in the figure.

Suppose that we are given the following preference instances as training data: P_Q(A,B), P_Q(B,C), P_Q(C,D), and P_Q(D,E). Under the standard approach to applying decision trees to numerical feature spaces described in [Quinlan, 1986], each point P in feature space would be represented as a 4-element vector $v \in \{0,1\}^4$, where the first element is 1 if $P > F1$ and 0 otherwise, the second element is 1 if $P > F2$ and

0 otherwise, *etc.,* and where features F1, F2, F3, and F4, represent the midpoints of the intervals between feature values of each point. For example, point A would be represented as $< 1, 1, 0, 0 >$, since A is greater than F1 and F2, but less than F3 and F4. Similarly, B would be represented as $< 1, 1, 1, 0 >$. Preference instance P_Q(A,B) would therefore be represented as $< A, B, + > \equiv < 1, 1, 0, 0, 1, 1, 1, 0, + >$ and $< B, A, - > \equiv < 1, 1, 1, 0, 1, 1, 0, 0, - >$. The five training instances would then be represented as shown in Figure 4.

	F1	F2	F3	F4	F5	F6	F7	F8	classification
P_Q(A,B)	1	1	0	0	1	1	1	0	+
P_Q(B,C)	1	1	1	0	1	0	0	0	+
P_Q(C,D)	1	0	0	0	1	1	1	1	+
P_Q(D,E)	1	1	1	1	0	0	0	0	+
P_Q(B,A)	1	1	1	0	1	1	0	0	−
P_Q(C,B)	1	0	0	0	1	1	1	0	−
P_Q(D,C)	1	1	1	1	1	0	0	0	−
P_Q(E,D)	0	0	0	0	1	1	1	1	−

Figure 4: Representation of preference instances P_Q(A,B), P_Q(B,C), P_Q(C,D), and P_Q(D,E) for decision-tree induction.

If ID3 is run on these training instances, the decision tree shown in Figure 5 is obtained: This decision tree incorrectly classifies the pair (B,D) as −. In effect, ID3

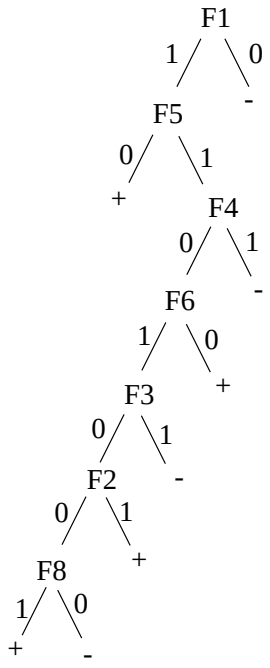

Figure 5: Decision tree generated by ID3.

has formed the generalization that B is less than anything except C and E. Nothing in the decision-tree induction process is able to exploit the inference from P_Q(B,C) and P_Q(C,D) that P_Q(B,D). Similarly, 1ARC is unable to make this inference. However, CIBL would correctly classify (B,D) by concatenating $\overleftarrow{CD}$ and $\overleftarrow{BC}$, shown in Figure 6. This example suggests that CIBL should be more accurate than decision-tree induction for quality functions having changes in the sign of their derivative.

Finally, a potential weakness of the instance-based approaches is that both depend on a Euclidean distance function. Instance-based methods that use Euclidean distance functions typically are sensitive to irrelevant features [Aha, 1989, Aha and Goldstone, 1990].

In summary, the characteristics of the arc representation of preference instances

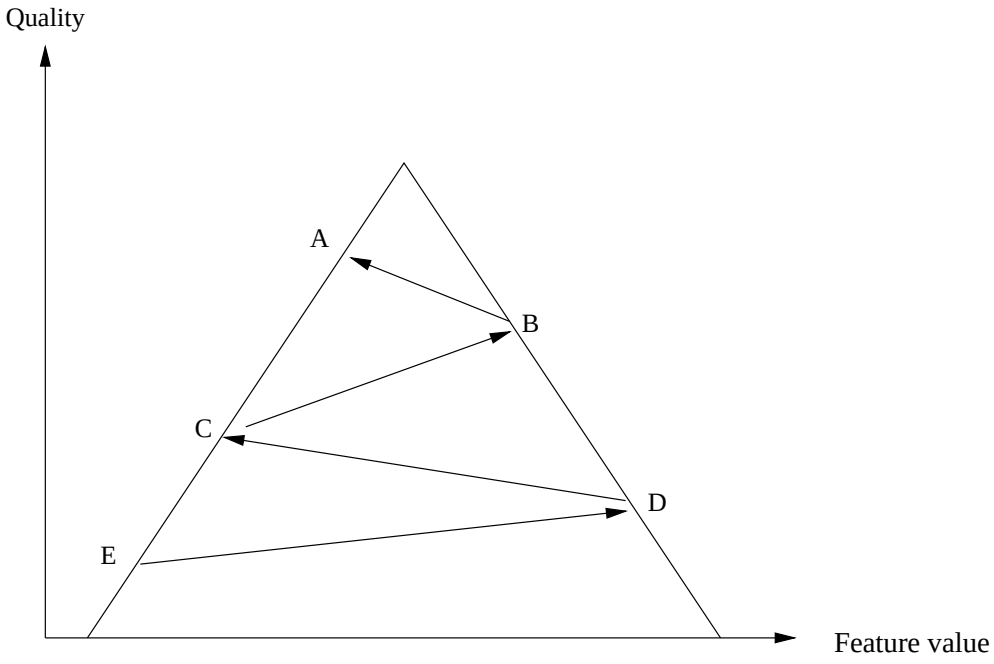

Figure 6: An arc representation of training instances P_Q(A,B), P_Q(B,C), P_Q(C,D), and P_Q(D,E). Concatenating $\overleftarrow{CD}$ with $\overleftarrow{BC}$ permits CIBL to conclude that P_Q(B,D).

lead to the following hypotheses:

1. CIBL outperforms 1ARC and decision-tree induction for quality functions with many derivative sign changes.

2. The state-preference method is much less accurate than the instance-based methods for quality functions that are poorly approximated by linear functions (although highly accurate for linear quality functions).

3. Introduction of irrelevant features lowers the accuracy of the instance-based methods more than the accuracy of decision-tree induction.

4. The instance-based approaches outperform decision-tree induction when there are small numbers of training instances.

The next section describes an empirical evaluation designed to test these hypotheses.

3 Empirical Evaluation

To test the hypotheses set forth in the previous section, the ranking accuracy of 1ARC and CIBL was compared to the accuracy of decision-tree induction and the state-preference method on a variety of artificial quality functions of varying "complexity", *i.e.,* varying numbers of derivative sign changes (see Figure 7). With the exception of Q_8, all of the quality functions were defined on the feature space $S = [0, 1] \times [0, 1]$. Functions Q_1 and Q_2 are linear functions, with identical and greatly differing feature importances, respectively. Q_3 is an exponential. Q_1 through Q_3 are all functions with

no changes is derivative sign. Q_4, a quadratic, Q_5, crossed planes, and Q_7, a 2-cycle sinusoid, all involve derivative sign changes in both features. Q_7, is the most complex function in the sense that it has multiple changes in derivative sign in each dimension. Q_6, a folded plane, has a derivative sign change in a single dimension.

The decision-tree induction method tested was ID3, modified to handle real-valued features in the manner described above. The state-preference method was tested using a standard implementation of perceptron learning. For each Q function, instances of the associated preference predicate, $P_Q(X, Y)$, representing the knowledge "X is preferred over Y" for $X, Y \in S$ were randomly generated.[3] Each model was trained on a set of instances of size $\|TS\| \in \{2, 8, 32, 128\}$ and was then tested on a different set of instances of size 1000. Each $< model, Q, \|TS\| >$ triplet was trained and tested four times and the mean error rate was calculated by counting the incorrect rankings in the four tests.

The accuracy of the learning methods with 128 training instances is summarized

[3]For ID3, each $P_Q(X, Y)$ was encoded as a positive and a negative instance of the concept "is preferred to" using a feature vector of real values. In addition to the standard features described above, ID3's instance representation included the normalized direction of the instance and its magnitude:

$$< +, Y, X, (X - Y)/\|X - Y\|, \|X - Y\| >$$
$$< -, X, Y, (Y - X)/\|X - Y\|, \|X - Y\| >$$

Encoding for perceptron learning was in the form of the pairs:

$$< +, Y, X >$$
$$< -, X, Y >$$

$Q_1(f_1, f_2)$	=	$f_1 + f_2$	1x1 plane
$Q_2(f_1, f_2)$	=	$f_1 + 10f_2$	1x10 plane
$Q_3(f_1, f_2)$	=	$exp(f_1^2 + f_2^2)$	exponential
$Q_4(f_1, f_2)$	=	$(f_1 - 0.5)^2 + (f_2 - 0.5)^2$	quadratic at (0.5,0.5)
$Q_5(f_1, f_2)$	=	$\begin{cases} f_2 & \text{if } f_1 \leq 0.5 \\ 1 - f_2 & \text{otherwise} \end{cases}$	crossed planes
$Q_6(f_1, f_2)$	=	$\begin{cases} f_1 + f_2 & \text{if } f_1 \leq 0.5 \\ 1 + f_2 - f_1 & \text{otherwise} \end{cases}$	folded plane
$Q_7(f_1, f_2)$	=	$sin(2\pi(f_1 + f_2))$	2-cycle sinusoid
$Q_8(f_1, f_2, f_3)$	=	$(f_1 - 0.5)^2 + (f_2 - 0.5)^2$	2-D quadratic in 3-D

Figure 7: Quality functions of varying complexity.

in Figure 8. Consistent with hypothesis 1, CIBL was more accurate than the other learning methods in quality functions that changed derivative sign in both dimensions, *i.e.,* in the quadratic (Q_4), crossed-planes (Q_5), and the 2-cycle sinusoid (Q_7). CIBL's accuracy was significantly higher than that of ID3 for the quadratic (P = .0066) and sinusoid (P = .0002), but the difference was not clearly significant for the crossed-planes (P = .0723). There was no significant difference in accuracy between CIBL and ID3 for the folded plane (Q_6), which changes derivative sign in only one dimension. CIBL was significantly more accurate than 1ARC on the quadratic (P = .0066), folded plane (P = .0221), sinusoid (P = .0021), and 2-dimensional quadratic with an added irrelevant feature (Q_8)(P = .0003). This provides confirmation for the hypothesis

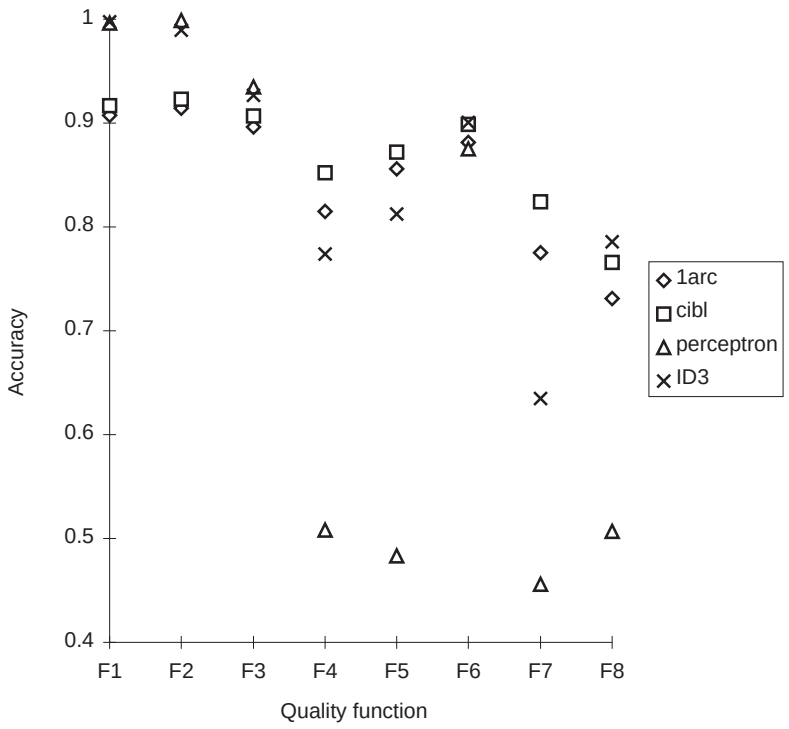

Figure 8: Accuracy of learning methods in learning Q functions of varying complexity.

that composing multiple instances can lead to improved accuracy. By contrast, ID3 was significantly more accurate than CIBL for the 1×1 plane (Q_1) and 1×10 plane (Q_2) (P = .0021 and P = .0041 respectively).

Hypothesis two, that perceptron learning is much less accurate than the instance-based methods for quality functions without accurate linear approximations but highly accurate for linear quality functions, was confirmed as well. Perceptron learning performed almost perfectly on the linear functions, Q_1 and Q_2, but no better than chance on the functions that vary in derivative sign in both dimensions, Q_4, Q_5, Q_7, and Q_8. There were no significant differences in accuracy among the learning methods for the exponential ($Q3$) and folded-plane ($Q6$).

The third hypothesis, that introduction of irrelevant features lowers the accuracy of the instance-based methods more than the accuracy of decision-tree induction, was confirmed by the 2-dimensional quadratic with an added irrelevant feature (Q_8). Although CIBL significantly outperformed ID3 in the quadratic with no irrelevant features (Q_4), the two methods were not significantly different when an irrelevant feature was added.

Hypothesis four is that the instance-based methods are more accurate than decision-tree induction given small numbers of training instances. This hypothesis is satisfied in functions that vary in derivative sign in both dimensions, because the instance-based methods outperform decision-tree induction for all training set sizes tested. More interesting are the linear quality functions. As shown in Figure 9 and Figure 10, with only two training instances the instance-based methods were significantly more

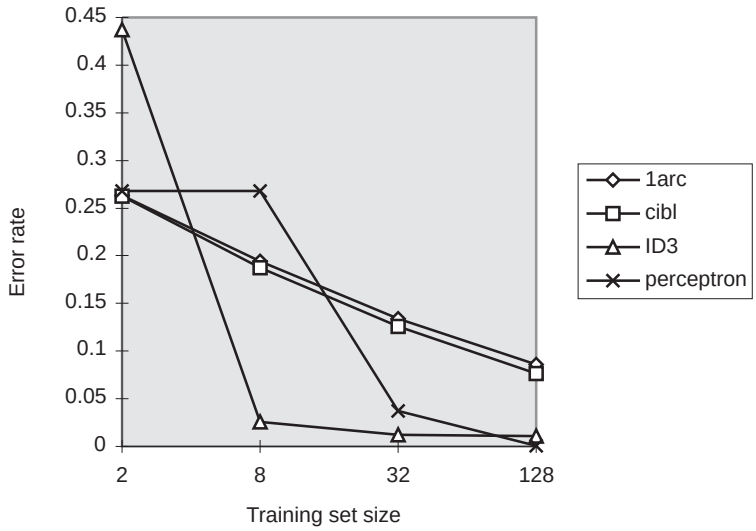

Figure 9: Error rate as a function of training set size for a 1×10 plane.

accurate than ID3 on both linear functions and more accurate than perceptron learning for the 1×1 plane. Given eight training instances, the instance-based approaches are more accurate than ID3 on the 1×1 plane and more accurate than perceptron learning for the 1×10 plane. However, for 32 or more training instances the error rate on the linear functions for ID3 and perceptron learning was negligible. Figure 11 shows that the instance-based methods were more accurate than the other methods on the folded plane given only two training instances, but the relative performance of the various methods converge with larger numbers of training instances.

Intuitively, one would expect that quality functions over feature spaces with larger numbers of dimensions would require more instances to learn than quality functions over feature spaces with fewer dimensions. This suggests that CIBL would perform

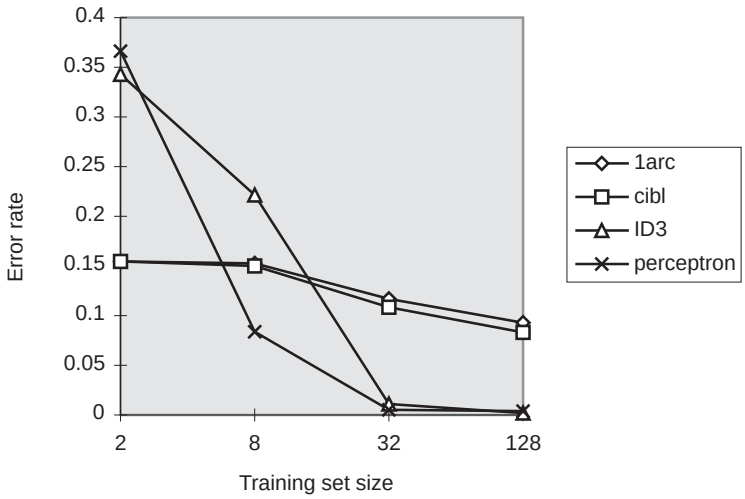

Figure 10: Error rate as a function of training set size for a 1×1 plane.

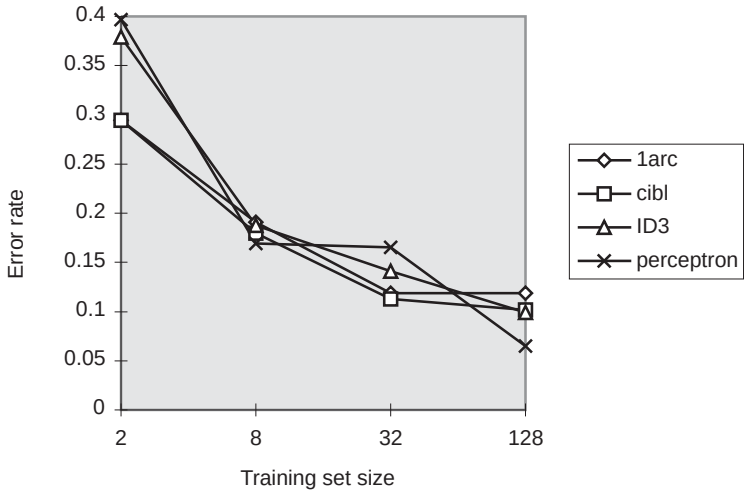

Figure 11: Error rate as a function of training set size for a folded plane.

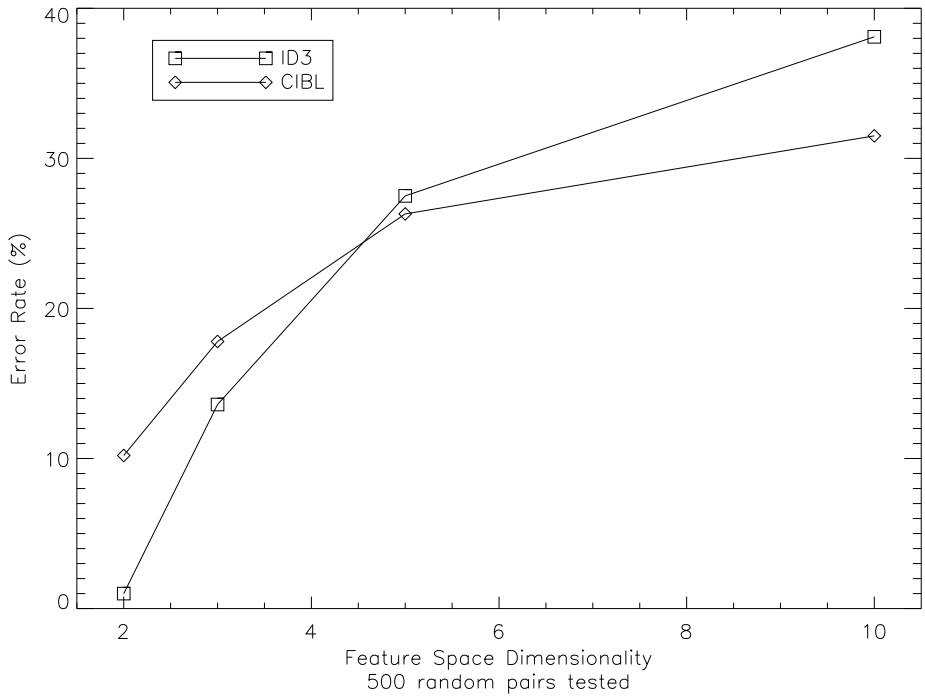

Figure 12: Cumulative error rate of CIBL and ID3 for linear Q in feature spaces of dimension 2, 3, 5, and 10.

better relative to decision tree induction as the number of dimensions increases. To test this hypothesis, the ability of CIBL and ID3 to acquire a preference predicate was compared for a linear Q over features spaces of dimensionality 2, 3, 5 and 10, with training set size of 128 and testing set size of 500. As with the earlier experiment, both training and testing instances were uniformly distributed through the feature space. The results, set forth in Figure 12, show that for linear Q ID3 has a lower error rate in feature spaces of dimensionality less than 5, the error rate is comparable for dimensionality equal to 5 and CIBL has a lower error rate in feature spaces of dimensionality greater than 5.

In summary, the results of the empirical evaluation provide initial confirmation for each of the hypotheses set forth in the previous section.

4 Acquisition of Preference Predicates for Astronomical Scheduling

This section describes a prototype application of preference learning in an advising system for astronomical scheduling. The purpose of the prototype is to illustrate the role of user preferences in configuration tasks typified by scheduling and to demonstrate the feasibility of automated preference acquisition in such tasks.

4.1 The Telescope Scheduling Task

At most ground-based observatories, each observer is responsible for planning the observations that are made during his or her allotted observing time. An experienced observer typically has a catalog of desired observations that far exceeds the allotted time. As a result, optimizing the use of an astronomer's limited observation time is essential to the astronomer's research productivity.

Astronomers typically construct schedules incrementally, starting with an empty schedule and adding one object at a time until the allotted observing time is filled. If the optimal placement of a new observation is inconsistent with some observations in the current partial schedule, the astronomer typically (1) performs a repair operation in which the inconsistent observations are moved slightly from their optimal placement

to accommodate the new observation, and/or (2) places the new observation in a non-optimal location.

In this process of incremental schedule construction, astronomers must necessarily make trade-offs among the various relevant schedule attributes. Astronomers typically have little uncertainty about the optimal value of each of these characteristics considered in isolation. For example, ignoring all other considerations, the airmass of each observation should be minimized. However, informal discussions with astronomers suggest that they cannot easily articulate the criteria they use in making these trade-offs. In particular, astronomers can neither specify a set of rules for choosing between any two proposed schedules nor formulate a merit function that measures the relative quality of schedules. Moreover, individual astronomers appear to differ significantly in their preferences.

Previous approaches to automated telescope scheduling [Johnston, 1989] [Barrett and Thomas, 1991] have cast the problem as a *constraint satisfaction problem* (CSP). The CSP framework provides a useful representation of hard constraints. Many such scheduling systems also provide facilities for encoding preferences and for using those preferences to guide a state space search [Zweben et al., 1992, Feldman and Golumbic, 1989, Freuder, 1989, Dhar and Ranganathan, 1990, Johnston, 1989]. For example, in SPIKE [Johnston, 1989], a system used to schedule observations on the Hubble Space Telescope, hard constraints and preferences are encoded as *suitability functions* taking on non-negative values. All the suitability functions are multiplied together to form a *total suitability function* that is used to choose between proposed schedules. However,

any such *a priori* encoding of preferences presupposes the exact tradeoffs that will be made between scheduling preferences.

4.2 The Observing Assistant Learning Apprentice

The Observing Assistant[4] is a decision support system that assists astronomers in scheduling ground-based telescope observations. Starting with an empty schedule, OA suggests refinements to the current partial schedule by adding one object from the astronomer's catalog of desired observations.

The set of possible refinements of a partial schedule, S, consists of each placement into S of an unscheduled object from the astronomer's catalog that results in a new schedule satisfying all hard constraints. In general, there are a very large number of possible refinements since there are many new observations that could be added, each new observation could be scheduled at almost any time and, indeed, the observations already in the schedule could be rearranged in many ways. OA uses two methods to limit the number of possible refinements it considers in order to reduce computation time in an interactive environment and to avoid overloading the user with choices.

First, when a refinement is created by inserting an unscheduled object into S, the order of the observations already in S is not altered. Second, once an unscheduled object has been inserted into S, the times of all the observations are adjusted to minimize the mean airmass while preserving the order of the observations. This scheme does not emulate all of the scheduling strategies employed by humans. For example,

[4] A more detailed description of the Observing Assistant is set forth in [Broos, 1993].

humans may employ complex backtracking steps such as switching the position of two objects already in the schedule. A general-purpose schedule editor in OA allows the human to make complex changes to the schedule by hand.

After a set of refinements to the current partial schedule are generated, OA uses its model of the astronomer's preference predicate to sort the set. The highest ranked refinement is then suggested to the user. If the user rejects the proposed refinement s_i in favor of some other refinement s_j, OA records the ordered pair (s_j, s_i) to represent the fact that a preference for s_j over s_i is an instance of the user's preference predicate, $P_Q(s_j, s_i)$.

For the purposes of ranking alternative schedules each schedule is represented as a vector of the following real-valued attributes:[5]

- the duration of the newest observation added
- the maximum airmass of the newest observation
- the optimal airmass of the newest observation, *i.e.,* the lowest airmass achieved by the newest object during the entire night
- the priority of the newest observation
- the average airmass of the other objects in the schedule.

An empty schedule is represented as a set of time segments representing periods when observing is possible, each of which is specified by its start time and length.

[5]Several additional attributes, such as total telescope slew time, would need to be added for a complete model of the factors considered by astronomers in scheduling.

Multiple time segments permit the scheduling of several nights simultaneously if desired and permit the representation of planned idle periods during a single night. A catalog of objects that can be scheduled is given in the form of a simple table whose format was designed to be compatible with the catalogs currently in use by the telescope-aiming computer at the Wyoming Infrared Observatory.[6]

Suppose the current schedule S contains three objects A, B, and C.

Schedule S: .....aa.bbb....cc... *dots represent gaps*

Suppose also that the objects remaining in the catalog are X, Y, and Z. Then OA will put the twelve schedules below into the set *Refinements*.

.XXX.aa.bbb....cc...	.....Yaabbb....cc...	......ZZabbb...cc...
..aaXXX.bbb....cc...	.....aaYbbb....cc...	.....aa.ZZbbb..cc...
.....aabbbXXX..cc...	.....aa.bbb..Y.cc...	.....aa.bbb...ZZcc..
.....aa.bbbccXXX....	.....aa.bbb...ccY...	.....aa.bbb...ccZZ..

The gaps between objects (represented by dots) are the result of the airmass minimization repair step. Once OA has generated the possible refinements shown above it groups the refinements that involve the same new object and ranks each group. For

[6]Each object is represented in this format as follows:

- Name of the object
- Right ascension coordinate
- Declination coordinate
- Duration of observation
- Priority (HIGH, MEDIUM, or LOW).

example, the refinements shown above would be grouped into three queues and each would be sorted using OA's preference function model.

QUEUE FOR X	QUEUE FOR Y	QUEUE FOR Z
1 .XXX.aa.bbb....cc...	1 .....aa.bbb..Y.cc...	1 .....aa.bbb...ZZcc..
2 ..aaXXX.bbb....cc...	2 .....aa.bbb...ccY...	2 .....aa.bbb...ccZZ..
3 .....aabbbXXX..cc...	3 .....aaYbbb....cc...	3 .....aa.ZZbbb..cc...
4 .....aa.bbbccXXX....	4 .....Yaabbb....cc...	4 ......ZZabbb...cc...

Once the best placement for each new object is known (the top row above), those best placements are ranked with respect to each other, forming a top-level queue as shown below. The top ranked refinement in the top-level queue is OA's estimate of the best possible refinement to schedule S.

TOP-LEVEL-QUEUE
1 .....aa.bbb..Y.cc...
2 .....aa.bbb...ZZcc..
3 .XXX.aa.bbb....cc...

When a two-level set of queues containing schedule refinements is constructed as described above, OA allows the user to browse the queues, correcting OA's rankings if desired. The user may browse through the top-level queue, examining the best placement of each new object, or he may browse one of the placement queues, examining all the possible ways to add a specific new object.

When browsing any queue, the entries in the queue are presented to the user in pairs to facilitate comparison. The pair of entries displayed consists of a *marked entry* and a *browse entry* (see Figure 13). The marked entry is the one the user currently deems to be the best one in the queue. The browse entry is the one the user is currently comparing to the marked entry. The browsing menu has commands NEXT and PREV to change the browse entry to the next or previous entry in the queue.

The command MARK will assign the role of marked entry to the schedule currently displayed as the browse entry. When the queue is first created, the marked entry is the schedule that is ranked highest by OA.

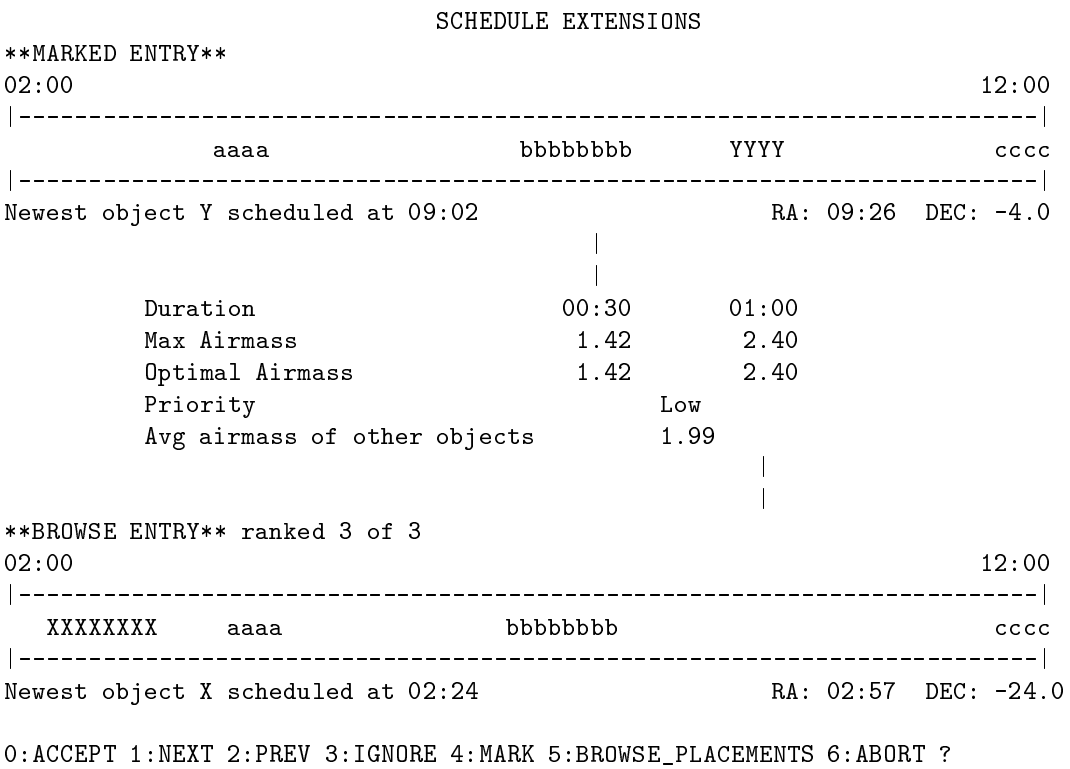

Figure 13: Browse Display

When the user leaves the top-level queue with the ACCEPT command, the schedule in the marked entry display is accepted as the desired refinement to the current schedule. When the user leaves the lower-level queue with the ACCEPT command, the schedule in the marked entry display is substituted into the top-level queue. OA's preference learning component obtains a training pair each time the user corrects OA's

ranking of a queue by accepting a queue entry that was not top-ranked by OA.

4.3 Scheduling Experiments with Astronomers

4.3.1 Interactive Learning Experiment

The experiments using artificial quality functions indicated that CIBL always performs at least as well as 1ARC and that the relative performance of CIBL and ID3 depends upon the nature of the underlying quality function Q, the dimensionality of the feature space, and the number of training instances. The second set of experiments compared the relative effectiveness of CIBL to that of ID3 on the task of learning an astronomer's scheduling behavior in the context of the Observing Assistant. Two different versions of OA were implemented: OA-CIBL, which used the CIBL learning method; and OA-ID3, which used the ID3 learning method.

A typical observing catalog of astronomical objects was provided by the director of the Wyoming Infrared Observatory at the University of Wyoming. An astronomer at the University of Wyoming Department of Physics and Astronomy then scheduled this catalog twice, once using OA-CIBL and once using OA-ID3. The catalog comprised three nights of observations to be scheduled, so a total of six nights were scheduled (three nights per catalog, two different learning methods). The six learning sessions were interleaved so that the astronomer did not know which learning method was in use. Each time the astronomer made a ranking decision, that is, each time the astronomer expressed a preference for a particular schedule in a set of schedules, data were collected on OA's ranking of the schedule the astronomer preferred and the

number of schedules the astronomer had to choose from.

The relative performance of the learning algorithms was measured in two different ways. The first measure of performance was cumulative error rate, which indicates how often each model failed to identify correctly the astronomer's preferred schedule. The cumulative error rate of CIBL was lower (37%) than that of ID3 (47%), but the difference was not statistically significant.

The second measure of performance was a *linear pay-out metric*. If two preference models, A and B, both rank a set of 30 schedules and model A assigns a rank of 3 to the human's preferred choice and model B assigns a rank of 27, it is reasonable to say that model A performed better than model B even though neither model predicted the human's preferred choice. A linear pay-out metric is a more appropriate measure of performance in situations in which a near miss is nearly as good as a correct answer. We used a linear pay-out metric that rewards a model by $\frac{2(n-m)}{n-1} - 1$ if the model assigns the user's first choice out of n objects a rank of m. This metric rewards a scheduler by $+1.0$ when the user's chosen schedule was ranked first and by -1.0 when the user's chosen schedule was ranked last. The expected value of this metric for a preference predicate model with no knowledge is zero. Figure 14 shows the cumulative pay-out data for OA-CIBL and OA-ID3, indicating that both had about the same ability to predict the astronomer's behavior. The relatively high pay-out from both methods—over 40 after 62 instances—indicates that both methods rapidly acquired

a sufficiently accurate preference model to provide useful advice to the astronomer.[7]

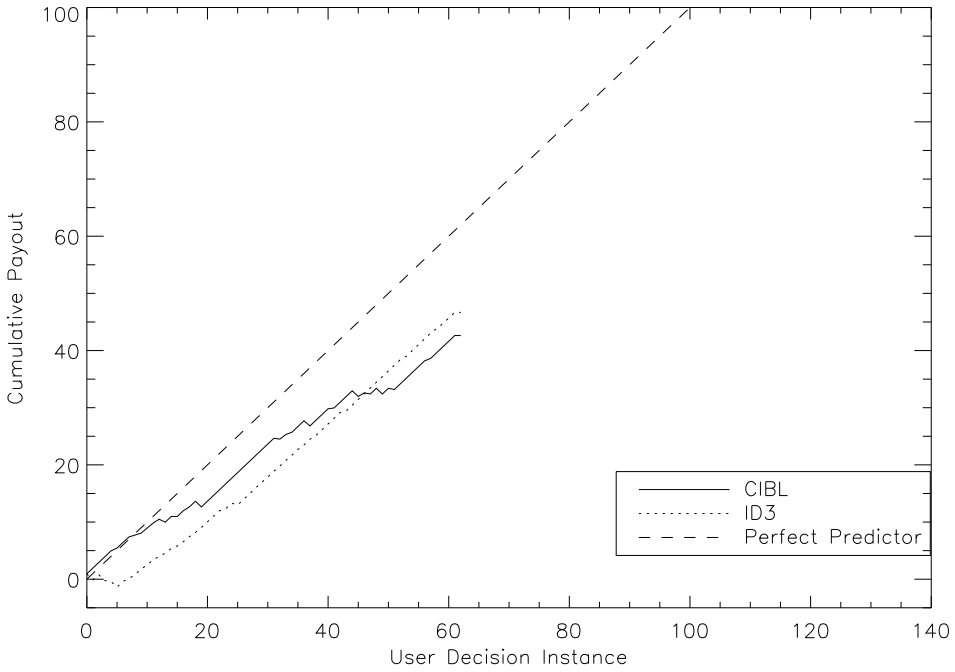

Figure 14: Cumulative pay-out. The 45° line represents the cumulative payout of a perfect model of the astronomer's preference predicate.

4.3.2 Replay Experiments

In addition to directly measuring the relative performance of CIBL and ID3 as the learning component of OA, the learning methods were compared on two sets of approximately 135 preference instances recorded from each of two different astronomers

[7]The slightly higher pay-out for OA-ID3, notwithstanding its somewhat lower accuracy, indicates that the average magnitude of errors was somewhat greater for OA-CIBL.

who used OA to schedule 6 nights of observations. These instances are examples of the astronomers' preference predicates.

In the first experiment, each astronomer's preference instances were used to train CIBL and ID3 separately using a learn-on-failure protocol. The two states contained in each preference instance were given to the model (CIBL or ID3) for ranking, and the model learned the instance only if it ranked the states incorrectly. The cumulative error rates of the two models (astronomer #1: CIBL-21%, ID3-22%; astronomer #2: CIBL-14%, ID3-21%) were not significantly different, confirming the result of the interactive learning experiment described above that CIBL and ID3 had comparable abilities to predict astronomer's behavior.

The second replay experiment tested the hypothesis that different astronomers use distinct preference predicates. The two sets of preference instances were each randomly partitioned into two subsets. One partition was used to train a preference predicate model. The model's error rate was then measured on the task of predicting the preferences contained in the other partitions. This experiment was performed under twelve different configurations to cover all the possible permutations of three configuration variables: the preference model used (CIBL or ID3); the source of the training partition (astronomer #1 or astronomer #2); and the size of the training partition (45, 68, or 90 instances). The experiment was repeated 10 times for each testing configuration.

Over all 120 tests, the average error rate for ranking instances from the set used to train the model (8.7%) was significantly lower than the average error rate for ranking

instances from the other astronomer's set (25.0%) (P = .00137). This indicates that there was a significant difference between the scheduling behaviors of the two astronomers we tested, confirming the hypothesis that different astronomers require different preference models.

4.4 Scheduling Experiments Using Artificial Preference Predicates

The final experiment tested whether the dependence of the relative performance of CIBL and ID3 on the complexity of the underlying quality function Q, which was observed in an artificial domain, also applies when scheduling actual astronomical observations. To test this hypothesis, OA-CIBL and OA-ID3 were rerun on the catalog of observations using each of the quality functions set forth in Figure 15 as an oracle in place of a human astronomer. As shown in Figure 16, the results confirmed that CIBL's performance relative to ID3 improves with increasingly complex Q: ID3 is more accurate than CIBL for linear Q,[8] CIBL is slightly more accurate for quadratic Q, and CIBL is much more accurate for sinusoid Q.

[8]This result appears to be inconsistent with the artificial domain experiment that tested the effect of dimensionality on ranking accuracy, in which ID3 and CIBL had comparable accuracy for linear Q in a five-dimensional feature space. However, this disparity is attributable to the differences between the two experiments: (1) the instances used for training and testing were random points in feature space for the earlier experiment but were actual schedules for the later experiment and (2) the task in the earlier experiment was to establish a binary ranking whereas the task in the later experiment was to order a full set of schedule refinements.

Q_9	=	$f_1 - f_2 + f_3 + f_4 - f_5$	5-D plane
Q_{10}	=	$-[(f_1 - 1)^2 + (f_2 - 2)^2 + (f_3 - 1.5)^2$	
		$+(f_4 - 0.5)^2 + (f_5 - 2)^2]$	5-D quadratic
Q_{11}	=	$sin(\pi\sqrt{f_1^2 + f_2^2 + f_3^2 + f_4^2 + f_5^2})$	5-D sinusoid

Figure 15: Quality Functions.

In summary, the Observing Assistant illustrates how the preference acquisition task can arise in the context of a learning apprentice system. The interactive learning experiments illustrated that preference predicate acquisition techniques can rapidly acquire models of user preferences sufficiently accurate to provide useful advice, and the second replay experiment provided evidence that different astronomers do in fact apply different quality functions.

5 The Impact of Representation on Preference Predicate Acquisition

The empirical evaluation indicates that the relative performance of instance-based and inductive approaches to preference predicate acquisition depends on the dimensionality of the feature space, the number of training instances, and the complexity of the preference predicate P_Q being acquired as measured by the underlying evaluation function Q. However, the nature of Q depends critically on the representation of instances.

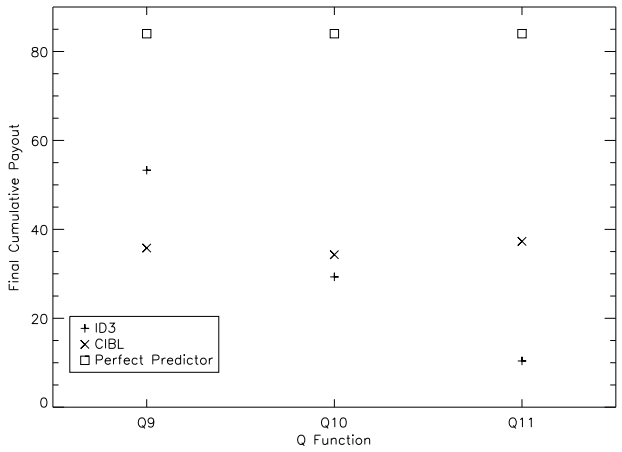

Figure 16: Cumulative pay-out of CIBL and ID3 with 5-D plane, quadratic, sinusoid functions replacing the human astronomer.

If instances are represented in terms of raw observables, any quality function on those instances is likely to be extremely irregular. For example, if chess positions are represented purely in terms of the locations of each piece on the board, the evaluation function will be extremely irregular, because changing the position of a single piece can drastically alter the evaluation of the position. If instances are represented in terms of *derived* or *abstract* features, however, the evaluation function may become much smoother. For example, if chess positions are represented in terms of strategic features such as control of files or pawn structure, or in terms of tactical features such

as the existence of pins or the security of the king, an incremental change in a feature will usually change the evaluation function only incrementally. The ideal instance representation for the acquisition of preference predicates would include the quality function Q itself as a derived feature.[9]

Thus, a quality function that is a highly irregular when applied to a low abstraction representation may become smooth when applied to a higher abstraction representation. This suggests that a key issue in choosing between instance-based and inductive approaches to acquisition of preference predicates is the nature of the representation of the instances. If instances are represented at a low level of abstraction, an instance-based approach like CIBL may be superior because of the irregularity of the quality function when applied to such descriptions. Induction may be more appropriate if instances can be represented in terms of more abstract features.

6 Conclusion

Acquisition of user preferences is important for a significant class of advising tasks, typified by configuration tasks such as design or scheduling. Learning apprentice acquisition of preference predicates is an appropriate technique for learning user preferences when (1) users can identify the relevant characteristics of problem-solving

[9] A well-known illustration of the dependence of inductive learning techniques on the representation of instances is Quinlan's experience that devising a set of derived features for chess board positions sufficient to enable ID3 to induce a decision tree for "lost in 3-ply" required 2 person-months [Quinlan, 1983].

states, (2) these state characteristics can be adequately represented as an attribute vector, but (3) users differ as to or are unable to articulate evaluation criteria for problem solving states in terms of these attributes.

The scheduling experiments involving artificial preference criteria, the relative performance of the preference predicate learning methods was found to depend on (1) the complexity of the preference predicate P_Q being acquired as measured by the underlying evaluation function Q, (2) the dimensionality of the feature space, and (3) the number of training instances. CIBL's strategy of composing multiple instances always led to accuracy equal to or higher than that of 1ARC. CIBL appears preferable to ID3 as the learning component of a learning apprentice system if representation of the relevant characteristics of problem-solving states requires more than five attributes or if attributes interact in a complex fashion, *i.e.*, if the quality function has derivative sign changes in multiple dimensions, provided that all attributes are relevant. Conversely, ID3 is preferable if there are fewer than five attributes and the attributes do not interact in a complex fashion (*i.e.*, the quality function has few derivative sign changes) or if there are irrelevant attributes. Perceptron learning performed extremely well for quality functions with accurate linear approximations, but extremely poorly for quality functions with derivative changes in multiple dimensions. Finally, the instance-based methods performed better than ID3 or perceptron learning for extremely small numbers of training instances (for extremely small numbers of training instances 1ARC and CIBL are equivalent).

The scheduling experiments with astronomers indicated that both CIBL and ID3

can be effective as the learning component of a learning apprentice system for acquisition of preference predicates. Both CIBL and ID3 rapidly acquired a useful level of accuracy when tested as the learning component of a learning apprentice used by an astronomer for scheduling astronomical observations having five real-valued attributes.

Acknowledgments

This research was supported in part by NASA Space Grant Graduate Fellowships administered through the Wyoming Planetary and Space Science Center. The implementations of ID3 and perceptrons used in this research were by Ray Mooney.

Appendix A

Proof that a preference instance parallel to the gradient of a linear quality function will correctly classify all instances with respect to the quality function under the 1ARC procedure.

Let $Q(x_1, \ldots, x_n) = \sum_{i=1}^{n} a_i x_i$ be a linear quality function of n features. The gradient of Q is the vector G=$< a_1, \ldots, a_n >$. A preference instance parallel to G in features space must be of the form P_Q(A,B) $\equiv (< u_1 + ca_1, \ldots, u_n + ca_n >< u_1, \ldots, u_n >)$, where c is a positive constant. Let (W,Z) $\equiv (< w_1, \ldots, w_n >< z_1, \ldots, z_n >)$ be a testing instance. 1ARC ranks W and Z by finding whether $\overleftarrow{AB}$ more closely matches

$\overleftarrow{WZ}$ or $\overleftarrow{ZW}$. The distance between $\overleftarrow{AB}$ and $\overleftarrow{WZ}$ is

$$dist(A,W) + dist(B,Z) = \sum_{i=1}^{n}[(u_i + ca_i - w_i)^2 + (u_i - z_i)^2]$$

Similarly, the distance between $\overleftarrow{AB}$ and $\overleftarrow{ZW}$ is

$$dist(A,W) + dist(B,Z) = \sum_{i=1}^{n}[(u_i + ca_i - z_i)^2 + (u_i - w_i)^2]$$

Thus, 1ARC will rank W as preferable to Z only if

$$\sum_{i=1}^{n}[(u_i + ca_i - w_i)^2 + (u_i - z_i)^2] < \sum_{i=1}^{n}[(u_i + ca_i - z_i)^2 + (u_i - w_i)^2]$$

However, this inequality can be simplified to

$$\sum_{i=1}^{n} w_i a_i > \sum_{i=1}^{n} z_i a_i$$

which is equivalent to $Q(W) > Q(Z)$.

Similarly, 1ARC will rank Z as preferable to W only if

$$\sum_{i=1}^{n}[(u_i + ca_i - w_i)^2 + (u_i - z_i)^2] > \sum_{i=1}^{n}[(u_i + ca_i - z_i)^2 + (u_i - w_i)^2]$$

which can be simplified to

$$\sum_{i=1}^{n} w_i a_i < \sum_{i=1}^{n} z_i a_i$$

which is equivalent to $Q(W) < Q(Z)$. Thus, the single training instance P_Q(A,B) correctly classifies all testing instances with respect to Q.

References

[Aha, 1989] Aha, D. (1989). Incremental, instance-based learning of independent and graded concepts. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 387–391.

[Aha, 1990] Aha, D. (1990). *A Study of Instance-Based Algorithms for Supervised Learning Tasks.* PhD thesis, University of California at Irvine.

[Aha, 1992] Aha, D. (1992). Generalizing from case studies: A case study. In *Proceedings of the Ninth International Workshop on Machine Learning*, pages 1–10.

[Aha and Goldstone, 1990] Aha, D. W. and Goldstone, R. L. (1990). Learning attribute relevance in context in instance-based learning algorithms. In *Twelfth Annual Conference of the Cognitive Science Society*, pages 141–149.

[Aho et al., 1974] Aho, A., Hopcroft, J., and Ullman, J. (1974). *The Design and Analysis of Computer Algorithms.* Addison-Wesley Publishing Co.

[Barrett and Thomas, 1991] Barrett, J. D. and Thomas, R. C. (1991). An algorithm for the support of telescope microscheduling. *Publications of the Astronomical Society of the Pacific*, 103:1218–1230.

[Broos, 1993] Broos, P. (1993). An expert system for telescope scheduling. Master's thesis, University of Wyoming.

[Callan et al., 1991] Callan, J., Fawcett, T., and Rissland, E. (1991). Adaptive case-based reasoning. In *Proceedings of the Third DARPA Case-Based Reasoning Workshop*, pages 179–190. Morgan Kaufmann.

[Dent et al., 1992] Dent, L., Boticario, J., McDermott, J., Mitchell, T., and Zabowski, D. (1992). A personal learning apprentice. In *Proceedings of Tenth National Conference on Artificial Intelligence*, pages 96–103, San Jose, CA. AAAI Press/MIT Press.

[Dhar and Ranganathan, 1990] Dhar, V. and Ranganathan, N. (1990). Integer programming vs expert systems: an experimental comparison. *Communications of the ACM*, 33(3):323–336.

[Feldman and Golumbic, 1989] Feldman, R. and Golumbic, M. (1989). Constraint satisfiability algorithms for interactive student scheduling. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1010–1015.

[Freuder, 1989] Freuder, E. (1989). Partial constraint satisfaction. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 278–283.

[Johnston, 1989] Johnston, M. D. (1989). Knowledge based telescope scheduling. In Heck, A. and Murtaghn, F., editors, *Knowledge-Based Systems in Astronomy*. Springer-Verlag.

[Maes and Kozierok, 1993] Maes, P. and Kozierok, R. (1993). Learning interface agents. In *Proceedings of Eleventh National Conference on Artificial Intelligence*, pages 459–466, Washington, D.C. AAAI Press/MIT Press.

[Mahadevan et al., 1993] Mahadevan, S., Mitchell, T., Mostow, J., Steinberg, L., and Tadepalli, P. (1993). An apprentice-based approach to knowledge acquisition. *Artificial Intelligence*, 64(1).

[Mitchell et al., 1985] Mitchell, T., Mahadevan, S., and Steinberg, L. (1985). LEAP: A learning apprentice for VLSI design. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann.

[Quinlan, 1983] Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess end games. In Carbonell, J. G., Michalski, R. S., and Mitchell, T. M., editors, *Machine Learning*, volume 1, pages 463–482. Tioga, Palo Alto, CA.

[Quinlan, 1986] Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1:81–106.

[Quinlan, 1993] Quinlan, R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

[Stanfill and Waltz, 1986] Stanfill, C. and Waltz, D. (1986). Toward memory-based reasoning. *Communications of the ACM*, 29(12).

[Utgoff and Clouse, 1991] Utgoff, P. and Clouse (1991). Two kinds of training information for evaluation function learning. In *Proceedings of Ninth National Conference on Artificial Intelligence*, pages 596–600, Anaheim. AAAI Press/MIT Press.

[Utgoff and Saxena, 1987] Utgoff, P. and Saxena, S. (1987). Learning a preference predicate. In *Proceedings of the Fourth International Workshop on Machine Learning*, pages 115–121.

[Zweben et al., 1992] Zweben, M., Davis, E., Daun, B., and Deale, M. (1992). Rescheduling with iterative repair. Technical Report FIA-92-05, NASA Ames Research Center.