Learning User Preferences in Distributed Calendar Scheduling

Jean Oh and Stephen F. Smith

School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
{jeanoh,sfs}@cs.cmu.edu

Abstract. Within the field of software agents, there has been increasing interest in automating the process of calendar scheduling in recent years. Calendar (or meeting) scheduling is an example of a timetabling domain that is most naturally formulated and solved as a continuous, distributed problem. Fundamentally, it involves reconciliation of a given user's scheduling preferences with those of others that the user needs to meet with, and hence techniques for eliciting and reasoning about a user's preferences are crucial to finding good solutions. In this paper, we present work aimed at learning a user's time preference for scheduling a meeting. We adopt a passive machine learning approach that observes the user engaging in a series of meeting scheduling episodes with other meeting participants and infers the user's true preference model from accumulated data. After describing our basic modeling assumptions and approach to learning user preferences, we report the results obtained in an initial set of proof of principle experiments. In these experiments, we use a set of automated CMRADAR calendar scheduling agents to simulate meeting scheduling among a set of users, and use information generated during these interactions as training data for each user's learner. The learned model of a given user is then evaluated with respect to how well it satisfies that user's true preference model on a separate set of meeting scheduling tasks. The results show that each learned model is statistically indistinguishable from the true model in their performance with strong confidence, and that the learned model is also significantly better than a random choice model.

1 Introduction

One vision of research in the field of intelligent software agents is the realization of personal computer assistants. A personal computer assistant is a software agent that is integrated into a user's computing environment and pro-actively accomplishes various tasks in support of high-level user goals. Like a human

assistant, such a personal computer assistant would do such things as process email, schedule meetings, service information requests, organize events, and so on; autonomously interacting with other personal computer assistants as necessary to carry out a given task and in each case recognizing if and when it is appropriate to engage the user in the process. One fundamental aspect of a personal computer assistant is that it is enduring and self-improving. It is expected to persist indefinitely, and learn over time to make decisions that better reflect user constraints and preferences.

Toward this goal of creating personal computer assistants, there has been increasing interest in automating the process of scheduling meetings and managing user calendars. Calendar scheduling can be seen as a kind of timetabling problem - the objective is to assign time slots to meetings in such a way that the constraints and preferences of meeting requests and prospective attendees are best satisfied. However, the problem of calendar scheduling differs from typical timetabling problems in a couple of important respects:

- Continuous, dynamic problem - Calendar scheduling is an ongoing endeavor. At any point in time, there are some number of meetings booked, and new requests are serviced continuously in an incremental manner. It is generally desirable to maintain stability in assignments over time, although invariably it will be necessary to bump previously scheduled meetings and the busier individuals are the more frequent such tradeoffs will need to be considered.
- Distributed decision-making - Although one can consider centralized approaches to the calendar scheduling problem, this requires all individuals involved to share their calendars and this is not a realistic assumption in many circumstances. In these cases, protocols for negotiating time slots that are mutually acceptable to prospective attendees must be devised. Note that in some situations it may be necessary to settle on a subset of attendees, and/or coordinate with additional resource brokers (e.g., room booking agents).

Like other timetabling domains, calendar scheduling preferences will vary from user to user, and one strong prerequisite of any calendar scheduling solution is an ability to incorporate user-specific preferences. Preferences can range from simple static time of day (or day of week) preferences, to more complex dynamic preferences (such as scheduling meetings back-to-back or retaining free time in proximity to important deadlines), to preferences of which meeting(s) to bump in over-constrained situations. Typical timetabling solutions require users to directly specify their preferences as input to the problem solving process. However the fact that calendar scheduling is an ongoing, continuous process suggests the possibility of automatically acquiring this knowledge over time through observation of meeting scheduling episodes.

Our recent research in the calendar scheduling domain has led to development of CMRADAR, a distributed calendar scheduling system[9]. Each CMRADAR

scheduling agent accepts requests for meetings from its user , and interacts autonomously with the CMRADAR agents of other users to determine and confirm a mutually agreeable meeting time. If the action of scheduling a given meeting preempts a previously scheduled meeting, then affected CMRADAR agents coordinate to reschedule the bumped meeting. CMRADAR meeting scheduling protocols support a range of negotiation strategies, allowing the amount of information exchanged (e.g., number of options, preference values), and the assumptions made about organizational structure to be varied. Scheduling options are proposed and evaluated by CMRADAR agents based on how well they satisfy underlying user preferences. CMRADAR uses a quantitative preference representation, (i.e., a user preference is specified as a utility function), and assigns a value indicative of degree of satisfaction to a given scheduling option. Since such a specification of preference is somewhat unnatural for the user whom a given CMRADAR agent is assisting, we consider the possibility of automating the acquisition of user preferences. Another advantage of learning preferences automatically is that statistical observations can reveal certain meaningful patterns that are often missed by the human users.

Our general hypothesis is that it is possible for a software agent to learn a user's meeting time scheduling preferences by observing the user engage in a series of meeting scheduling episodes with other meeting participants. In this paper we describe an initial proof of principle experiment. We make specific assumptions about the types of information that can be observed during a meeting scheduling episode and the organizational setting in which meeting scheduling takes place, and with these assumptions we investigate an approach to learning the user's true preference model. In particular, we assume that a learning software agent has access to the following information when observing the user schedule meetings: (1) the user's current calendar, (2) incoming and outgoing meeting requests (initiator, proposed time slots), (3) user replies (accept or refuse) and (4) confirmed meeting time slots. We further assume that meeting scheduling takes place within a hierarchical organization, that the learning agent has knowledge of the respective ranks of various meeting participants in the organization, and that users in the organization use a common negotiation strategy when scheduling meetings that favors the preferences of higher ranked individuals. Our specific hypothesis is that under these assumptions, accumulation of the above meeting information over some number of user scheduling episodes is sufficient to enable the agent to learn the user's true meeting time preference. To test this hypothesis we use a set of CMRADAR scheduling agents to simulate meeting scheduling under the above assumptions and generate training data for the learning agent. We then evaluate the ability of the learning agent to learn the true preference model of a given CMRADAR agent.

2 Related Work

There has been growing interest in creating software agents that can assist the users with daily routine tasks. In the particular problem of calendar scheduling there exist several commercial software calendar programs that support some form of basic meeting scheduling protocol. For instance, Microsoft Outlook Exchange Server provides capabilities such as finding intersections of free time slots of all attendees, and sending out meeting requests via automatically generated email messages, etc. Most commercial software rely on the protocol in which a centralized server has access to every individual calendar. In general meeting scheduling often takes place in more distributed settings where the members of an organization are not obliged to use a specific calendar software but elect to use their own choice of calendar programs. More fundamentally, these tools are limited in the fact that the user's scheduling preferences and habits are ignored.

Sen et al. [10] applied voting theory to their distributed scheduling agent system which tries to compromise in the event of conflicting user preferences during the negotiation process. The preferences are associated with utility values similar to our approach, but the user is responsible for specifying the quantitative data manually. Calendar Apprentice (CAP) [5][8] used the decision tree learning method to learn user preference rules. Blum [2] improved CAP's performance by applying Winnow and Weighted-Majority based algorithms. CAP suggests specific values for the attributes of meeting such as duration and time slot. For example, the time of day preference rule suggests a time slot for a given type of meeting. If the suggested time slot is already taken by another meeting the closest available time slot is suggested. On the other hand, CMRADAR learns a utility function to evaluate different alternatives. PCalM [1] is another system that learns an evaluation function, in this case using large margin method [6] and Naive Bayesian approach with additional active learning strategy. However, no experimental results have been reported with this system in the literature.

3 Basic Modeling Assumptions

For purposes of this paper, we adopt the following basic modeling assumptions:

- We define a calendar to be a sequence of time slots of equal duration over some horizon. Let C_u be the calendar of user u, and $C_u(t)$ refer to time slot t of C_u.
- A meeting request $Req_{i,A,T}$ is initiated by an initiator i, and designates a set of attendees A and a set of one or more proposed time slots T.
- A reply or response to meeting request r by user u is designated $Resp_{u,r}$ and specifies a value of either *accept* or $refuse$ for each time slot $t \in T$.

- A Scheduled Meeting $M_{i,A,t}$ similarly designates an Initiator i, a set of attendees A and a specific time slot t. For all $u \in A$, $C_u(t) = M_{i,A,t}$.
- A given time slot $C_u(t)$ of user u's calendar will either contain a scheduled meeting M or is *available*. Only one meeting may be scheduled at a given time slot. Hence for any new meeting M' to be scheduled at a given time slot t, either $C_u(t) = available$ or the currently scheduled meeting $M_{I,A,t} : u \in A$ must be bumped.
- A *static* user meeting time preference model is expressed as a utility curve over some sequence of time slots. A static time of day (TOD) preference is a utility curve over a sequence of TOD time slots (e.g., the sequence of slots 7AM, 8AM, ...6PM). A time of week (TOW) preference model would be specified similarly (albeit using a larger sequence of TOW time slots). The value associated with a given time slot t in a preference utility curve ranges from 1.0 (most preferred) to -1.0 (most negatively preferred), with 0 designating a tolerable lower bound on acceptability in peer to peer negotiation circumstances. Let $Pref_u(t)$ designate the preference value for time slot t by user u. When scheduling a meeting, $Pref_u$ determines the relative desirability of different *available* time slots.
- We assume a hierarchical organization with n levels. A given individual i in the organization has a rank R_i, where $1 \leq R_i \leq n$. Individuals with higher rank reside at higher levels in the organization.

From the standpoint of the learning agent, the goal is to observe user u in the process of scheduling meetings and to acquire u's preference curve $Pref_u$. We assume that the learner sees each meeting request $Req_{i,A,T}$ involving u, each $Resp_{u,r}$ involving u, and receives confirmation of each scheduled meeting $M_{i,A,t}$ involving u. The learner also has access to u's current calendar at all times. Finally, the learner has knowledge of the ranks of all individuals in the organization, and assumes that all individuals use a common strategy for negotiating meeting times in which the preferences of higher ranked individuals are favored.

4 Approach

We take a statistical approach to learning a static time-of-day (TOD) preference curve for a given user from observed meeting scheduling data. During a series of meeting scheduling episodes the learning agent observes the user's actions relative to the set of time slots in C_u. As meeting scheduling proceeds, the user proposes various time slots to initiate new meetings, accepts or refuses the time slots proposed by other users, and receives confirmations of mutually agreed upon meeting times. Conceptually, our approach views such actions and results as *noisy* examples of the user's underlying preference model (both positive and negative). In order to turn this observed data into a meaningful characterization of the user's time of day preference, we map the collected set of scheduling

actions and results into "votes" for each time slot. The key point of our approach concerns how to weight these votes to minimize the noise.

In more detail, the following information is collected as the user engages in meeting scheduling:

- TOD Time slots proposed by the user when initiating a meeting. In this case, proposed time slots provide active positive evidence of the user's true preference. Accordingly, we define $InitPropCt_u(t)$ to accumulate this positive evidence for different TOD time slots (e.g., 7 AM, 8 AM, etc.). The potential obscuring factor (or source of noise) in this data, however, is the density of C_u; if the most preferred time slots are already occupied, then less preferred time slots will necessarily be proposed. Taking this fact into account, each time the user proposes a given TOD time slot t when initiating a meeting, the following computation is performed:

$$InitPropCt_u(t) = InitPropCt_u(t) + (1 - Density_{C_u})$$

  where

$$Density_{C_u} = \frac{OccupiedSlots_{C_u}}{TotalSlots_{C_u}}.$$

  In other words, the evidence for a given proposed TOD time slot is discounted by the current density of C_u.
- TOD Time slots that are available but refused by the user when responding to a meeting request. In this case, the user's response provides active negative evidence for the time slot(s) in question. We define $RefusedCt_u(t)$ to accumulate this negative evidence for various time slots. Each time the user refuses a proposed TOD time slot t that is actually available in C_u, the following computation is performed:

$$RefusedCt_u(t) = RefusedCt_u(t) + 1.$$

- TOD Time slots of confirmed meeting times. These time slots also can provide passive positive evidence of the user's true preference (since the user has agreed to this time slot). As in the case of those time slots proposed by the user when initiating a meeting , $Density_{C_u}$ can be an obscuring factor and must be discounted. However, here there is also a second source of noise relating to the relative ranks of meeting attendees in the organization. Taking into account the fact that all users employ a common negotiation protocol that favors higher ranked individuals, we assume that the user will tend to reveal more truthful preferences when negotiating with lower ranked individuals. To account for this, evidence relating to confirmed meeting times is differentiated by the rank of the meeting initiator. Specifically, we define a matrix $ConfirmedCt_u(r, t)$, and each time a TOD time slot t proposed by an individual of rank r is confirmed as a meeting time, the following update is performed:

$$ConfirmedCt_u(r, t) = ConfirmedCt_u(r, t) + (1 - Density_{C_u}).$$

Using the above computations, we collect "votes" for each TOD time slot. We then use the weighted k-nearest neighbor (KNN) algorithm [4] to consolidate this data and smooth the curve. KNN was initially proposed by Fix and Hodges [7]. It is a popular statistical approach which has been used heavily in the pattern recognition research and also in the text categorization. The basic idea here is to predict the utility value for a given TOD time slot using k similar data points in the training set. Here similarity is defined as a combination of both distance between TOD time slots and the distance between a meeting initiator's rank and the user's rank in the organization. The influence of a given data point on another is discounted as a function of its distance.

In more detail, the learned user preference model is computed according to the following four step procedure:

1. **Integrate TOD time slot values in** $ConfirmedCt_u$ - Taking the user's rank R_u into account, weighted KNN is applied to average the values accumulated for each TOD time slot (i.e., each column in the matrix). KNN is applied asymmetrically in this case using rank distance as a similarity metric. Specifically, values accumulated for meetings initiated by individuals of rank $R_i > R_u$ are increasingly discounted as rank distance increases, based on the above stated intuition that meetings initiated by higher ranked individuals give less information. On the other hand, values for meetings initiated by individuals of rank $R_i \leq R_u$ are given full weight, since the user's preference will dominate in this case. The result of this smoothing step is a flattened vector of Confirmed meeting votes, designated $FlatConfirmedCt_u$.
2. **Combine collected data** - Compose final "votes" for each TOD time slot as follows:

$$TS_u = w_1 \times InitPropCt_u + w_2 \times FlatConfirmedCt_u - w_3 \times RefusedCt_u$$

3. **Smooth adjacent time slot data** - Weighted KNN is applied again, this time to the consolidated TOD time slot vector TS_u. Following the assumption that the actual (true) user preference will tend to be continuous, each TOD time slot value is averaged with the values of the k neighboring TOD time slots, discounted by TOD distance.
4. **Normalize final values** - Finally, the values in TS_u are normalized to the range $[-1, 1]$ to produce the learned preference utility curve.

In the experiment described below, the above procedure is invoked after observing a fixed number of meeting scheduling episodes. Note however, that algorithm could be applied to compute (and recompute) the utility curve dynamically as more data points are accumulated over time.

5 Evaluation

Our evaluation contrasts the performance of three preference models:

- **true model** - the preference model used to simulate meeting scheduling and generate the training data used by learner
- **learned model** - the preference model generated by the learning agent.
- **random model** - a preference model that randomly assigns utility values to time slots.

Each of these models is evaluated with respect to the true model - i.e., how well each model approximates the scheduling outcomes produced by the true model. Each model is applied to schedule a common (new) sequence of meetings, and in each case the final resulting calendar is evaluated with respect to how well it satisfies the true user preference model. More precisely, the quality of the resulting schedule is determined as:

$$Q = \sum_{m \in MTGS_u} \frac{Pref_u(TimeSlot(m))}{|MTGS_u|}$$

, where $MTGS_u$ is the set of meetings in C_u and $TimeSlot(m)$ is the time slot in which meeting m is scheduled.

6 Experimental Design

We use a set of CMRADAR agents to simulate meeting scheduling in a 4 person organization. In our experiment, this CMRADAR simulation serves two purposes. First, it is used to generate training data for learning a given user's preference model. Second, it is used to evaluate the performance of a learned user preference model during the test phase of the experiment. The simulation is configured as follows:

- Each CMRADAR "user" (agent) is given a unique preference curve. Experiments were run with two distinct sets of preference curves: (1) a *simple* configuration consisting of a combination of morning, afternoon, strictly morning, and strictly afternoon preferences, and (2) a *complex* configuration consisting of 4 randomly specified preference curves.
- We assume a 3-tiered organizational structure with $UserA > UserB > UsersC1, C2$.
- A common negotiation protocol that favors preferences of higher ranked individuals is utilized by all CMRADAR agents. Details of this negotiation protocol are given in Appendix A.

For purposes of the experiment user calendars are assumed to contain a total of 60 time slots per week; specifically a 5 day work week with each work day containing 12 one hour time slots from 7AM to 7PM. The target is to learn a daily TOD preference, i.e., a utility curve over the sequence of TOD time slots from 7AM to 7PM.

For the training (learning) phase of each experiment, 60 meeting requests to be scheduled over a two week horizon were randomly generated. 50 meetings were designated for week one and 10 for week two to ensure that some number of scheduling decisions must be made in the context of high density calendars. Each generated meeting involved 2, 3 or 4 of the individuals in the organization and the initiator was randomly assigned. Starting with empty calendars for all agents, the CMRADAR system is used to automatically schedule these 60 meetings. All meeting request messages, meeting response messages and meeting confirmation messages together with the users' evolving calendars are used along the way as training data for the learning algorithm summarized in section 4. Note that CMRADAR is a distributed system and each CMRADAR agent represents a different user in the organization. Hence, there are 4 different preference learners operating in this experiment, each associated with a specific CMRADAR agent (or user). Of course each learner only has access to the information that is local to its user.

For the test phase of each experiment, 20 new meetings to be scheduled in a one week interval were randomly generated as before. Each of 4 learned models was evaluated separately, by substituting one learned model for the corresponding true model in the CMRADAR system, and then running the system along with the true preference models for other CMRADAR users. For each revised configuration, the sequence of 20 new meetings was scheduled, starting again with empty calendars for all agents. By limiting the test case to 20 meetings, we ensure calendars that in the worst case are only $\frac{1}{3}$ full, and hence scheduled times are likely to be more reflective of user preference.

As indicated above, separate experiments were run for both simple and complex configurations of true preference models. For each configuration, 10 replications of the above training and test procedure were run using different meeting sets. The test results obtained using each different data set were averaged together to determine overall performance of the learned model. For comparison, we also computed average results obtained with the true and random models across all data sets.

7 Results

We analyze the results of both experiments relative to two basic hypotheses :

1. The performance of the learned model is comparable to the performance of the true model (i.e., the two models produce results that are statistically indistinguishable).
2. The performance of the learned model is indistinguishable from the performance of the random model.

Table 1 shows the experimental results obtained for the simple preference model configuration. It shows, for each of the 4 users, the quality of the final schedule produced by the learned model during the test phase from the standpoint of how well it satisfies the true preference model of each respective user (designated $Q_{Learned}$). The quality of the final schedules produced by the true model (Q_{True}) and the random model (Q_{Random}) are also shown, as well as a p-test evaluation of our basic hypotheses.

For all users, we see that there is sufficient evidence to reject the hypothesis that $Q_{Learned}$ is indistinguishable from Q_{Random}. In other words, $Q_{Learned}$ performs significantly better than Q_{Random} for all users. We note that the difference between $Q_{Learned}$ and Q_{Random} is smaller for users at lower ranks in the organization than it is for users at higher ranks (and this can be seen for the difference between Q_{True} and Q_{Random} as well). This reflects the fact that a common negotiation strategy favoring the preferences of higher ranked individuals is used, which reduces the overall influence of the preferences of lower ranked individuals.

Alternatively, it is not possible to reject the hypothesis that $Q_{Learned}$ is indistinguishable from Q_{True} for any user. To provide evidence in support of this hypothesis [3], confidence intervals for each pair of models were also computed. These are shown in Table 2. We can see with 95% confidence that there is strong evidence for $Q_{True} = Q_{Learned}$ for all users except user A. In each case, the confidence interval is found to contain zero and to be very small relative to the two scores. In the case of user A, $Q_{Learned}$ is actually found to be significantly *higher* than Q_{True}. The fact that $Q_{Learned} > Q_{True}$ is due to the fact that the learned preference model is stricter than the true model, which leads to increased pressure toward more preferred time slots in the true preference model. We would expect these Q values to balance out with additional training data. Figure 1 shows the true and learned preference models for each user from one of the simple preference configuration runs to give a graphical sense of their correspondence.

Table 3 shows the experimental results for the complex preference model configuration, and Figure 2 shows plots of learned and true models for each user from one of the experimental runs. As in the first experiment, the performance of the learned preference model is found to be better than that of the random model for all individuals in the organization (i.e., the hypothesis $Q_{Learned} = Q_{Random}$ is rejected in all cases). Again, it is the case that the hypothesis $Q_{Learned} = Q_{True}$ cannot be rejected for any user and computation of confidence intervals provides strong evidence that the hypothesis can be accepted. Table 4 shows the 95%

User	Q_{True}	$Q_{Learned}$	Q_{Random}	$Q_{True} = Q_{Learned}$ (p-value)	$Q_{Learned} = Q_{Random}$ (p-value)
A	0.882	0.889	0.200	Cannot Reject ($0.712 > 0.05$)	Very Strong Reject ($0.0000 < 0.01$)
B	0.936	0.938	0.864	Cannot Reject ($0.534 > 0.05$)	Very Strong Reject ($0.0068 < 0.01$)
C1	0.822	0.807	0.720	Cannot Reject ($0.347 > 0.05$)	Strong Reject ($0.020 < 0.5$)
C2	0.791	0.792	0.726	Cannot Reject ($0.505 > 0.05$)	Weak Reject ($0.061 < 0.1$)

Table 1. Performance results for *Simple* preference model configuration.

User	Q_{True} - $Q_{Learned}$ interval	Width	$Q_{True} = Q_{Learned}$
A	(-0.028768,-0.006232)	0.022536	Cannot Accept (but $Q_{Learned}$ is better)
B	(-0.0070086,0.0028886)	0.0098971	Accept
C1	(-0.00086878,0.031358)	0.032226	Accept
C2	(-0.011393,0.010319)	0.021713	Accept

Table 2. 95% Confidence Intervals for *Simple* preference model configuration.

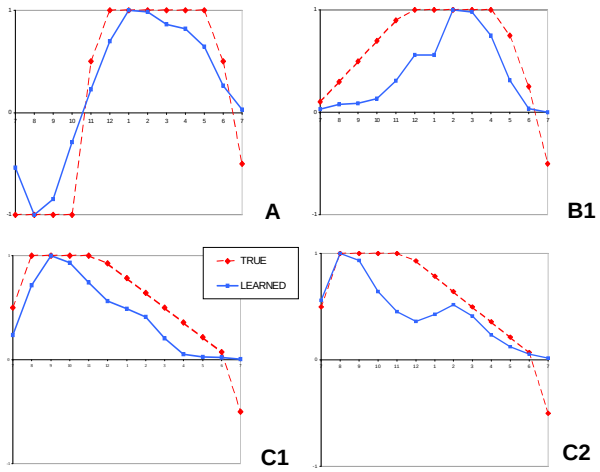

Fig. 1. True (Dashed) and Learned (Solid) Preference Utility curves for Simple Preference experiment.

confidence intervals for the complex model. For users B, C1 and C2, the results indicate that there is no significant difference between Q_{True} and $Q_{Learned}$. Not only does the confidence interval of the difference contain zero in each case, but it is also very small relative to the two scores. (e.g., for user B, the confidence interval of the difference between Q_{True} and $Q_{Learned}$ is $\frac{.0225}{.757} = 3\%$ of Q_{True}). In the case of user A, the confidence interval failed to cover zero. However, even though the difference between the two scores is not really zero, it is no more than $\frac{.035}{.826} = 4\%$ of Q_{True} with 95% confidence, indicating that performance of A's learned model very closely approximates that of A's true model.

User	Q_{True}	$Q_{Learned}$	Q_{Random}	$Q_{True} = Q_{Learned}$ (p-value)	$Q_{Learned} = Q_{Random}$ (p-value)
A	0.826	0.792	0.533	Cannot Reject ($0.191 > 0.05$)	Very Strong Reject ($0.0000 < 0.01$)
B	0.757	0.759	0.647	Cannot Reject ($0.52 > 0.05$)	Very Strong Reject ($0.0069 < 0.01$)
C1	0.607	0.602	0.510	Cannot Reject ($0.462 > 0.05$)	Strong Reject ($0.032 < 0.5$)
C2	0.634	0.640	0.501	Cannot Reject ($0.551 > 0.05$)	Very Strong Reject ($0.0025 < 0.01$)

Table 3. Performance results for *Complex* preference model configuration.

User	Q_{True} - $Q_{Learned}$ interval	Width	$Q_{True} = Q_{Learned}$
A	(0.016814,0.052084)	0.03527	Cannot accept
B	(-0.01343,0.0091317)	0.022562	Accept
C1	(-0.011222,0.020452)	0.031674	Accept
C2	(-0.018553,0.0062601)	0.024814	Accept

Table 4. 95% Confidence Interval for *Complex* preference model configuration.

8 Conclusion and Future work

These two experiments provide initial evidence of the ability to learn static user preference models for meeting scheduling through observation. Our ongoing work is investigating the implications for user preference learning of other sets of assumptions about meeting scheduling protocols and organizational structures, as well as extension of user preference learning techniques to incorporate more complex, dynamic preferences (e.g., a preference for scheduling meetings back-to-back, a preference to bump less important meetings). Currently CMRADAR utilizes a passive learning approach only, but we plan to integrate our system with an intelligent user interface to enable active learning by collecting user's feedback. Since the user's feedback provides the true answers, the agent's learning

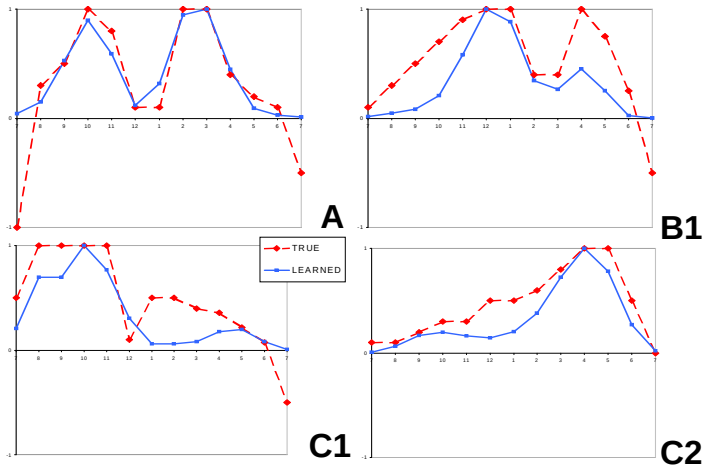

Fig. 2. True (Dashed) and Learned (Solid) Preference Utility curves for the Complex Preference experiment.

curve will be expedited. Knowing when and in what occasions the user should be interrupted is also another interesting learning task. Rule based strategies such as those proposed in CAP are more appropriate for capturing preferences more sensitively tuned to specific meeting types, i.e. implicitly recurring meetings. To tune our model to such customized meeting types, we anticipate extending our system to incorporate additional features in the distance metric, such as the subject of the meeting. If the number of features is large it will be more efficient to dynamically compute the utility values for alternative options, dynamically because the utility values will be customized according to the attributes of the given meeting by selectively choosing a subset of the data collected to date. For instance, if the given meeting is with person A and the subject of the meeting is Lunch, the system will use only the lunch meetings with person A to evaluate possible options, providing more customized evaluation criteria.

9 Acknowledgements

The authors thank Jay Modi and Manuela Veloso for the many stimulating discussions we have had on various aspects of calendar scheduling throughout the RADAR project. This research was sponsored in part by the Department of Defense Advanced Research Projects Agency (DARPA) under contract #NBCHC030029.

A Negotiation Protocols for CMRADAR Simulation

Within the CMRADAR simulation used in the experiment, each CMRADAR agent uses the following common negotiation protocol:

- **Initiator:** Initiator issues meeting request message and proposes a set of n options (time slots) that best suit its own preferences. In other words, the initiator proposes the n most preferred options (In the experimental runs $n = 3$.)
- **Attendees:** Each attendee responds to the meeting request as follows:
  - If one or more options are available, then evaluate and returned the *combined* preference value for each. (See below)
  - If there are no available options, *bump* the least important pre-emptable meeting, and return the combined preference value for this newly freed time slot.
- **Initiator:** Collect all attendee responses. If there is an agreeable option then confirm it. Otherwise, repeat above steps with n new options.

The above protocol implements a common policy of favoring the preferences of higher ranked individuals by communicating and combining the preference values of initiator and attendee. More precisely, Let

- $Pref_i(t)$ = the value assigned by Initiator i to time slot t and communicated to attendee A
- $Pref_a(t)$ = the value assigned by attendee a to time slot t
- $R_i and R_a$ = the ranks of i and a respectively

In evaluating a time slot proposed to attendee a by initiator i, a computes the following combined preference value:

$$Pref_{Comb}(t) = W_i \times Pref_i(t) + W_a \times Pref_a(t)$$

where

$$W_i = 0.5 + 0.5 * \frac{R_i - R_a}{R_{Max} - R_{Min}}$$

and

$$W_a = 1.0 - W_i$$

References

1. Pauline Berry, Melinda Gervasio, Tomas E. Uribe, Karen Myers, and Ken Nitz. A Personalized Calendar Assistant. *AAAI Spring Symposium Series, March*, 2004.

2. Avrim Blum. Empirical support for Winnow and Weighted-Majority based algorithms: results on a calendar scheduling domain. *Machine Learning*, 26:5–23, 1997.
3. Paul R. Cohen. *Hypothesis Testing and Estimation*, chapter 4. MIT Press, 1995.
4. Scott Cost and Steven Salzberg. A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features. *Machine Learning*, 10:57–78, 1993.
5. C. Lisa Dent, Jesus Boticario, John P. McDermott, Tom M. Mitchell, and David Zabowski. A Personal Learning Apprentice. In *Proceedings of AAAI-92*, pages 96–103, 1992.
6. Claude-Nicolas Fiechter and Seth Rogers. Learning Subjective Functions with Large Margins. In *ICML 2000*, pages 287–294, 2000.
7. E. Fix and J. L. Hodges. Discriminatory Analysis: Nonparametric Discrimination: Consistency Properties. Technical Report Project 21-49-004 4, USAF School of Aviation Medicine, Randolf Field, Texas, 1951.
8. Tom M. Mitchell, Rich Caruana, Dayne Freitag, John P. McDermott, and David Zabowski. Experience with a Learning Personal Assistant. *Communications of the ACM*, pages 80–91, 1994.
9. Pragnesh Jay Modi, Manuela Veloso, Stephen F. Smith, and Jean Oh. CMRadar: A Personal Assistant Agent for Calendar Management. In 6^{th} *International Workshop on Agent-Oriented Information Systems (AOIS)*, pages 134–148, 2004.
10. Sandip Sen, Thomas Haynes, and Neeraj Arora. Satisfying user preferences while negotiating meetings. *International Journal of Human-Computer Studies*, 47:407–427, 1997.