Machine Learning for Adaptive User Interfaces

Pat Langley

Intelligent Systems Laboratory
Daimler-Benz Research and Technology Center
1510 Page Mill Road, Palo Alto, CA 94304 USA
LANGLEY@RTNA.DAIMLERBENZ.COM

Abstract. In this paper we examine the growing interest in personalized user interfaces and explore the potential of machine learning in meeting that need. We briefly review progress in developing fielded applications of machine learning, then consider some characteristics of adaptive user interfaces that distinguish them from more traditional applications. After this, we consider some examples of adaptive interfaces that use inductive methods to personalize their behavior, and we report some ongoing research that extends these ideas in the automobile environment.

1 The Need for Personalized User Interfaces

Early computer software aimed to solve business and scientific problems in a predetermined way that allowed only very constrained user input, through arguments given to the program at run time. This contrasts sharply with modern-day software, which is much more interactive and supports frequent user input throughout its operation. This shift toward interactive software is reflected in the growing emphasis on interfaces designed to ease communication between software and humans.

Examples of such interactive software abound, and they have even become more common than the earlier, less interactive type, at least for nonspecialists. Most computer users have had experience with WYSIWYG editors for document preparation, with spreadsheets for handling financial data, with interactive computer games, and, most recently, with browsers and search engines for the World Wide Web. Moreover, there is every indication that the number, variety, and importance of such software will increase rather than decrease in years to come.

However, one major drawback of existing interactive systems is that they have little ability to take into account differences in the knowledge, style, and preferences of their users. Systems for document production let one select from a set of default styles and even store his own variations, but the latter process is manual and tedious. Computer games let one specify a difficulty level, but the opponent's strategy cannot reflect the user's strengths or weaknesses. Web browsers let one store bookmarks and preferred layouts, but search engines are only starting to incorporate user preferences to bias the retrieval process.

Clearly, there is a need for increased personalization in many areas of interactive software, not only in the types of flexibility but in the way that personalization occurs. To date, most systems have required that users state their

preferences explicitly to the interface, which means the options are either limited in number or tiresome to complete. Moreover, some facets of user styles may be reflected in their behavior but not subject to conscious inspection. This suggests the use of techniques from machine learning to personalize interfaces, based on the observation of user activity.

In the rest of this paper, we explore the potential of machine learning technology for automatic personalization of interactive software. We start by reviewing the state of machine learning and its recent successes in producing fielded applications, then discuss the characteristics of interactive software that differ from other types of learning applications. After this, we consider two broad categories of interactive software – informative and generative – and review some existing systems of each type that draw on machine learning in adapting to individual users. In closing, we consider some new research directions that we are pursuing to aid automobile drivers.

2 The Application of Machine Learning

Research on machine learning has existed since the beginnings of AI, and the past decade has seen considerable developments in this area. For the sake of discussion, we should define the field's object of study:

> A learning algorithm is a software system that improves its performance in some task domain based on partial experience with that domain.

This characterization of learning includes two important features that were absent from early work in the area. First, it states that the goal of learning is to improve performance on some task; the process may involve the creation of knowledge structures, but this is a means to the end of performance improvement. Second, it notes that learning involves *induction* beyond the training data, in that the system must perform after only partial experience with the task. Nearly all recent work on machine learning acknowledges these two requirements, typically reflecting them in their experimental tests of new algorithms.

However, machine learning has done more than evolve into a careful empirical science; it has also developed a successful applications methodology. Induction techniques have aided the construction of many fielded systems in science and industry on a variety of tasks. These include mechanical diagnosis (e.g., Giordana et al., 1993), credit scoring (e.g., Michie, 1989), manufacturing control (e.g., Evans & Fisher, 1994), and scientific classification (e.g., Fayyad, Smyth, Weir, & Djorgovski, 1995). Many of the standard induction algorithms have proven quite robust, and their increasing use for data mining and knowledge discovery promises even more successes in years to come.

The basic development process, although far from entirely automated, does cast machine learning in a central role (Brodley & Smyth, 1997; Langley & Simon, 1995; Rudström, 1995). Briefly, the developer works with a domain expert or user to understand some problem, then reformulates the problem into one solvable by well-established methods for supervised learning. He then selects some

likely features to describe training cases and devises an approach to collecting and preparing data, on which he runs some induction method. The developer (and possibly the expert) then evaluate the resulting knowledge base and, if the result seems acceptable, they attempt to deploy the learned knowledge in the field.[1]

Most academic work on machine learning continues to focus on refinements in induction techniques and downplays the steps that must occur before and after their invocation. Indeed, some research groups still emphasize differences between broad classes of induction methods, despite evidence that decision-tree techniques, connectionist algorithms, case-based methods, and other schemes often produce very similar results. In contrast, there is an emerging consensus within the applied community that the steps of problem formulation, representation engineering, and data preparation play a role at least as important as the induction stage itself. Indeed, there is a common belief that, once they are handled, the particular induction method one uses has little effect on the outcome (Langley & Simon, 1995).

We will adopt this viewpoint in our discussion of machine learning's potential for adaptive user interfaces. As a result, we will have little to say about the particular learning methods one might use to personalize an interface, but we will have comments about the nature of the performance task, the source of training data, and similar issues. This bias reflects our belief that adaptive user interfaces are an important application area for machine learning, and that strategies which have proved successful in other areas will also serve us well there.

3 The Nature of Adaptive User Interfaces

We can define adaptive interfaces by direct analogy with our definition for machine learning, using a slightly more specific formulation:

> An adaptive user interface is an interactive software system that improves its ability to interact with a user based on partial experience with that user.

As we suggested above, work on adaptive interfaces has much in common with other applied work on machine learning, including a reliance on careful problem formulation and engineering of useful features. For this reason, we will consider here only those characteristics that are special to this class of problems.

One central feature involves the manner in which the system uses the learned knowledge. Some work in applied machine learning is designed to produce expert systems, that is, knowledge bases (with associated performance elements) intended to replace a human. In contrast, work on adaptive interfaces aims to construct *advisory* systems, in which the knowledge base (through its performance element) only makes recommendations to the user. Rather than replacing

[1] Of course, this process is iterative, with problems at any step leading the developer to revisit earlier decisions.

a human, the system suggests information or generates actions that the user can always override. Ideally, the learned knowledge should reflect the preferences of individual users, thus providing personalized services for each one.

However, this focus on advisory systems leads directly to another characteristic – the user's decisions give a ready source of training data to support learning. Every time the interface suggests some choice, the human either accepts that recommendation or rejects it, whether this feedback is explicit or simply reflected in the user's behavior. Either way, the system obtains another datum to drive its search for an improved knowledge base, and each case includes details about the decision-making situation, providing important context for future predictions. This scenario contrasts with the situation for some potential applications of machine learning, where collecting data is a major obstacle.

The embedded nature of the induction process has another implication for the learning task: the system should carry out *online* learning, in which the knowledge base is updated each time an interaction with the interface occurs. This contrasts with most work in data mining, which assumes that all data are available at the outset. Because adaptive user interfaces collect data during their interaction with humans, one naturally expects them to improve during that use, making them 'learning' systems rather than 'learned' systems. This is not a strict requirement, in that the interface could collect data during a session, run the induction method offline, and then incorporate the results into the knowledge base before the next session, but the online approach seems the most natural.

Because adaptive user interfaces must learn from observing their user's behavior, another distinguishing characteristic is their need for *rapid* learning. The issue here in not CPU time, but rather the number of training cases needed to generate good advice. Most work on data mining assumes large amounts of data, typically enough to induce accurate knowledge bases even when the model class includes many parameters. In contrast, adaptive interfaces rely on a precious resource – the user's time – which makes the available data much more limited. This recommends the use of learning methods that achieve high accuracy from small training sets over those with higher asymptotic accuracy but slower learning rates. On the other hand, the advisory nature of these systems makes this desirable but not essential; an interface that learns slowly will be no less useful than one that does not adapt to the user at all. Still, adaptive interfaces that learn rapidly will be more competitive, in the user's eyes, than ones that learn slowly.

We can identify two broad classes of adaptive user interfaces, whose differences have implications for the type of feedback the user must provide. *Informative* interfaces attempt to select or filter information for the user, presenting only those items he will find interesting or useful. The most obvious examples are systems for product recommendation and news filtering, but this category includes any interface that directs the user's attention within a large space of items. Typical user feedback for informative systems includes marking recommended choices as desirable or undesirable, rating them on some scale, or giving some similar form of evaluation. Less obtrusive feedback can sometimes be col-

lected by observing the access process itself, as when one clicks on some items retrieved by a search engine but not others.

The second class, *generative* interfaces, focuses on the generation of some useful knowledge structure. Examples of this category include document preparation and drawing packages, spreadsheet programs, and systems for planning, scheduling, and configuration. These areas support richer types of feedback, in that the user can not only override a recommended action but can replace it with another one entirely.[2] The types of feedback are tied directly to the interaction styles that the interface supports. Some systems require the user to explicitly correct undesirable actions, but others incorporate less obtrusive schemes that collect training data simply by observing the user's normal behavior.

Now that we have considered the characteristics that distinguish adaptive user interfaces from other learning systems and the types of feedback they require, we can consider some examples of systems within both categories.

4 Examples of Informative Interfaces

The growing popularity of the World Wide Web has made informative interfaces the most familiar type of advisory system, and also the most common area for personalization. As their name suggests, such interfaces aim to provide the user with material that he will find informative or useful. Let us examine some existing systems that fall into this category.

One example interface is Pazzani, Muramatsu, and Billsus' (1996) SYSKILL & WEBERT, which recommends web pages on a given topic that the user is likely to find interesting. Starting from a handcrafted page for the topic, the user marks suggested pages as desirable or undesirable, and the system uses these as training data to develop a model of his preferences. SYSKILL & WEBERT incorporates a common scheme, known as *content-based* filtering, as the basis for selection and learning. Briefly, this approach represents each object using a set of descriptors, typically the words that occur in a document. The system uses these descriptors as predictive features when deciding whether to recommend a document to the user, which biases it toward documents that are similar to ones the user has previously ranked highly. Content-based methods also predominate in the older literature on information retrieval.

Another example is FILMFINDER, an interactive system that recommends movies one might enjoy. The user rates a number of sample movies, from which the system (available at `www.filmfinder.com`) constructs a simple user profile. FILMFINDER then finds other people with similar profiles and suggests films that they liked but that the current user has not yet rated. One can contrast this approach, known as *collaborative filtering*, with content-based methods, since it requires no prespecified descriptions of the objects or products being recommended. In effect, collaborative methods classify users (who are described in

[2] The term *learning apprentice* was originally used in the context of such generative systems, although some now use it to describe informative systems as well.

terms of the ratings they provide) rather than classifying the objects being recommended. This makes them well suited for subjective domains like art, where users base their decisions on intangible features that are difficult to measure.

A third system, WiseWire, which resides at `www.wisewire.com`, recommends news stories and web pages to its customers. This interface derives from Lang's (1995) NewsWeeder, which used a content-based method, but the new system combines content-based and collaborative filtering to select promising items. The intuition is that content-based methods are best for suggesting topics similar to ones the user has liked in the past, whereas collaborative methods can suggest items outside the user's normal area that he will still find interesting. WiseWire users rate the items that it recommends for inspection, but it also lets them note high-level reasons for their evaluation, which constrains the induction process and should improve the rate of learning.

This sample far from exhausts the list of interfaces for information filtering that incorporate machine learning to personalize their interaction with users. Other examples include systems for sorting electronic mail and for matchmaking among users with similar interests, and the number of applications is certain to grow as the Internet and its associated information sources become available to more and more people.

5 Examples of Generative Interfaces

Although informative systems are becoming familiar entities, they are not the only type of adaptive user interfaces. In some domains, one needs more than just information; one needs generative systems that actually construct new knowledge structures to satisfy the user's goals. Let us examine a few systems that address such tasks.

Hinkle and Toomey (1994) describe Clavier, an advisory system that recommends loads and layouts for aircraft parts to be cured in an autoclave. The system retrieves previous loads and their layouts from a case library, preferring ones that include more parts that are currently needed and that have cured well in past runs. A graphical interface presents a proposed load and layout to the expert user, who can then replace some parts or rearrange their positions. Each such modification provides a new case for the library, so that Clavier's repertoire grows over time. The system has been in continuous use since 1990, generating two to three autoclave loads per day and nearly eliminating problems due to incompatible loads. Although the Clavier effort did not focus directly on personalization, the system is a fine example of an adaptive interface of the generative variety.

Another example of an interactive generator comes from Hermens and Schlimmer (1994), who developed an adaptive system for filling out repetitive forms. Their interface suggests values for various fields in the form, but these are defaults that the user can always override. Once the user completes the form, the system interprets the entries as opportunities for learning and uses them to revise its existing rules. Each such rule predicts a default value for a given field

based on fields earlier in the form and those in previous forms. Experiments showed that the system reduced keystrokes by 87 percent over an eight-month period. Although this work did not focus on personalization per se, Schlimmer and Hermens (1993) took a very similar approach in their personalizing interface for note taking. This adaptive system learns a grammar that predicts the order and content of a user's notes, aiming to reduce keystrokes and to help them organize their thoughts.

Dent et al.'s (1992) CAP also aims directly at personalization issues, in this case trying to mimic a secretary's expertise at scheduling meetings for a professor. The system includes actions for specifying the day, time, duration, and location of a meeting, for which it offers default values. Again, the user can accept or replace these suggestions, with each decision providing data for the learning process. Induction occurs every night rather than in true online fashion, with CAP learning a separate set of rules for each action that it needs to predict. The system was in use regularly for some years by a departmental secretary to schedule a faculty member's meetings.

Work on *programming by demonstration* also aims to construct personalized generative interfaces by observing the user's behavior. For example, Cypher (1991) describes EAGER, a system that learns iterative procedures from observation in a HyperCard setting, then highlights the actions it anticipates for the user's approval. In general, systems in this paradigm focus on constrained tasks that support online learning from very few training cases, but they carry out induction every bit as much as ones that use less domain-specific learning algorithms. A collection by Cypher (1993) contains a representative sample of work on programming by demonstration.

Some efforts on intelligent tutoring systems also draw on machine learning with the aim of personalizing instruction. For instance, Langley and Ohlsson (1984) adapted methods for learning search-control knowledge to model individual student errors in arithmetic. Also, Anderson's (1984) technique of *model tracing* relies on careful observation of student behavior in the same way as generative user interfaces, giving advice only when the student diverges from acceptable paths. More recently, Baffes and Mooney (1995) have adapted techniques for theory revision to develop personalized student models using similar trace data.

6 Open Issues in Adaptive Interfaces

Although a variety of research and development efforts have shown the potential of adaptive user interfaces, there remains much room for extending their flexibility and their interaction style. In closing, we will consider two ongoing projects designed to address drawbacks in existing systems for personalized interfaces, both carried out at the Daimler-Benz Research and Technology Center and both concerned with improving the automobile environment. [3]

[3] Collaborators on these projects include Afsaneh Haddadi, Bryan Johnson, Annabel Liu, Seth Rogers, and Jeff Shrager.

The first effort aims to develop an informative user interface that is more interactive in nature, so that it communicates with the user rather than simply filtering items. The performance task involves recommending places of interest, such as restaurants or theaters, that the user might want as his destination. The system, which we call the *Adaptive Place Advisor*, starts by accepting a query like "Where should I eat?" in a restricted form of natural language. The advisor responds with its own questions, which are designed to refine the user's desires and, eventually, to arrive at a single place that meets his constraints.

The advisor draws on background knowledge, stored in a number of concept hierarchies, to direct this refinement process. For instance, in response to the above query, the system might ask "How about Thai food?" or "Would you like to take something out?". The user can answer these questions in the affirmative or negative, but he can also state a preferred alternative, such as "Let's have Chinese instead" or "I'd like a sit-down meal". The place advisor would continue the process, asking questions such as "Would you like Szechuan?" and using the answer to further narrow the options, until it has jointly agreed with the user on a unique place.

When the Adaptive Place Advisor first interacts with a user, it knows nothing about the person's preferences, so its guesses at each level are poor and the communication process is inefficient. However, as the system gains experience with a user, it collects statistics about his preferences and should come to suggest options he finds attractive, thus reducing the need for interaction. Of course, this is an empirical claim that we must still test with human subjects, but positive results with simpler recommendation systems give us reason for optimism. We must also extend the advisor to generalize beyond individual choices, so that it can predict the user's preferences for new options based on features they share with familiar ones, and improve the system's discourse model, so that its interaction with the user seems as natural as possible.

Generative user interfaces typically support richer interaction with the user than informative systems, but they often require special actions on the user's part. A second project in our group involves a generative interface that draws on special information sources to remain as unobtrusive as possible. Here the performance task is to generate a route between a driver's current and desired location that he will find attractive. This system, which we call the *Adaptive Route Advisor*, uses a Global Positioning System (GPS) to infer the driver's location and constructs routes by searching for paths through a digital map. The system biases this search using learned knowledge about the routes known to the driver, on the assumption that people usually prefer routes with familiar segments over those with unfamiliar segments. This is another empirical claim that we must test in experiments, but it seems plausible enough to use in our prototype software.

We have described the route advisor's learning methods in detail elsewhere (Rogers, Langley, Johnson, & Liu, 1997), so we will not focus on them here. Briefly, the system matches traces of the driver's past trips against segments in the digital map, then transforms them into a constrained context-free gram-

mar that describes his route knowledge at varying levels of resolution. From the driver's perspective, the important point is that the system obtains this personalized route knowledge from GPS traces of his trips, and thus requires no actions on his part other than normal driving behavior. We feel that this approach to unobtrusive data collection can serve as a role model for future work in generative adaptive interfaces, as more sophisticated sensors make similar schemes possible for a wider range of domains.

Clearly, we have only started to explore the space of adaptive user interfaces, and there certainly exist other ways in which we can improve them. But these improvements will only come from designing and implementing prototypes for particular domains, running experimental studies with human subjects, and evaluating their ability to personalize themselves to the user's needs. Although many factors will be important in this process, machine learning seems certain to play a central role.

References

1. Anderson, J. R. (1984). Cognitive psychology and intelligent tutoring. *Proceedings of the Sixth Conference of the Cognitive Science Society* (pp. 37–43). Boulder, CO: Lawrence Erlbaum.
2. Baffes, P. T., & Mooney, R. J. (1995). A novel application of theory refinement to student modeling. *Proceedings of the Thirteenth National Conference of the American Association for Artificial Intelligence* (pp. 403–408). Portland, OR: AAAI Press.
3. Brodley, C. E., & Smyth, P. (1997). Applying classification algorithms in practice. *Statistics and Computing*, *7*, 45–56.
4. Cypher, A. (1991). EAGER: Programming repetitive tasks by example. *Proceedings of CHI* (pp. 33–39). New Orleans: ACM.
5. Cypher, A. (Ed.). (1993). *Watch what I do: Programming by demonstration*. Cambridge, MA: MIT Press.
6. Dent, L., Boticario, J., McDermott, J., Mitchell, T., & Zaborowski, D. (1992). A personal learning apprentice. *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 96–103). San Jose, CA: AAAI Press.
7. Evans, B., & Fisher, D. (1994). Overcoming process delays with decision-tree induction. *IEEE Expert*, *9*, 60–66.
8. Fayyad, U. M., Smyth, P., Weir, N., & Djorgovski, S. (1995). Automated analysis and exploration of image databases: Results, progress, and challenges. *Journal of Intelligent Information Systems*, *4*, 1–19.
9. Giordana, A., Saitta, L., Bergadano, F., Brancadori, F., & De Marchi, D. (1993). ENIGMA: A system that learns diagnostic knowledge. *IEEE Transactions on Knowledge and Data Engineering*, *KDE-5*, 15–28.
10. Hermens, L. A., & Schlimmer, J. C. (1994). A machine-learning apprentice for the completion of repetitive forms. *IEEE Expert*, *9*, 28–33.
11. Hinkle, D., & Toomey, C. N. (1994). CLAVIER: Applying case-based reasoning to composite part fabrication. *Proceedings of the Sixth Innovative Applications of Artificial Intelligence Conference* (pp. 55–62). Seattle, WA: AAAI Press.

12. Lang, K. (1995). NewsWeeder: Learning to filter news. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 331–339). Lake Tahoe, CA: Morgan Kaufmann.
13. Langley, P., & Ohlsson, S. (1984). Automated cognitive modeling. *Proceedings of the Fourth National Conference of the American Association for Artificial Intelligence* (pp. 193–197). Austin, TX: Morgan Kaufmann.
14. Langley, P., & Simon, H. A. (1995). Applications of machine learning and rule induction. *Communications of the ACM*, *38*, November, 55–64.
15. Michie, D. (1989). Problems of computer-aided concept formation. In J. R. Quinlan (Ed.), *Applications of expert systems* (Vol. 2). Wokingham, UK: Addison-Wesley.
16. Pazzani, M., Muramatsu, J., & Billsus, D. (1996). Syskill & Webert: Identifying interesting web sites. *Proceedings of the Thirteenth National Conference of the American Association for Artificial Intelligence* (pp. 54–61). Portland, OR: AAAI Press.
17. Rogers, S., Langley, P., Johnson, B., & Liu, A. (1997). Personalization of the automotive information environment. *Proceedings of the Workshop on Machine Learning in the Real World: Methodological Aspects and Implications* (pp. 28–33). Nashville, TN.
18. Rudström, A. (1995). *Applications of machine learning*. Licentiate thesis, Department of Computer and Systems Sciences, Stockholm University, Sweden.
19. Schlimmer, J. C., & Hermens, L. A. (1993). Software agents: Completing patterns and constructing user interfaces. *Journal of Artificial Intelligence Research*, *1*, 61–89.

This article was processed using the LaTeX macro package with LLNCS style.