Interacting with learning agents: Implications for ML from HCI

Ian H. Witten, Craig G. Nevill-Manning

Department of Computer Science, University of Waikato,
Hamilton, New Zealand
{ihw,cgn}@cs.waikato.ac.nz

David Maulsby

CAMIS (Medical Information Section), Stanford University,
Palo Alto, California
maulsby@camis.stanford.edu

1. INTRODUCTION

Computers excel at repetitive tasks. But automating them usually involves programming, which is beyond the reach of most non-specialist users. One solution is for machines to learn procedures by observing users at work—and if this enhanced users' productivity and sense of achievement, they might even be persuaded to help the system by supplying some additional information. In principle, combining machine learning with instructional interaction should increase the speed with which tasks are acquired, and enhance reliability too.

This meeting of learning and interaction is often called "programming by demonstration" (PBD). Although many PBD systems have been exhibited (Cypher, 1993), they tend to place more emphasis on interaction than on learning. The problem is that mainstream machine learning schemes are not designed to deal with the additional constraints imposed by the interactive environment, nor capitalize on the extra resources it provides. In general, ML schemes:

- process all instances at once—whereas interactive systems provide examples incrementally, and expect instantaneous predictions;
- are designed to perform well on dozens, hundreds, or thousands of examples—whereas users are loath to provide more than two or three before seeing useful predictions;
- make predictions for every example—whereas in an interactive situation it is often better to use discretion and contribute few, accurate, predictions rather than include many incorrect ones;
- expect a sufficient number of examples to differentiate between relevant and irrelevant features—whereas a learning agent may have access to an unlimited number of features, but very few examples;
- expect the feature space to remain consistent between examples—whereas when the user's actions modify the workspace, some features may become irrelevant and new ones may emerge;
- are often unable to take advantage of domain knowledge and hints from the user—whereas it is only this information that enables learning from so few examples;

- expect the learning task to be clearly defined: what the instances are, what the goal is, when the task begins and ends—whereas users do not provide this information explicitly; it must be inferred by observing their actions;
- ignore sequential aspects—whereas the order in which actions occur is often crucial, and it is not always possible to recast tasks in a nonsequential manner;
- operate in isolation—whereas in PBD it is suicidal to ignore the user's feedback, and other information (such as predictions made by classifiers with different biases, and by sequence learning schemes) must be taken into account too;
- operate within a life-cycle that involves expert analysis of preliminary results, and careful evaluation of accuracy—whereas a concept learnt in PBD may be obsolete as soon as the task is completed, and thus not merit human analysis and reflection.

Of course, there are isolated counterexamples to each point, but we maintain that no ML system adequately accounts for the change of emphasis that intelligent agents require.

This paper analyzes the problems and provides suggestions for their solution. The next section reviews the major issues raised by learning agents from the point of view of ML, and section 3 discusses aspects of the interactive situation that can be used to provide additional leverage for learning.

2. ISSUES FOR MACHINE LEARNING

We identify five aspects of the interactive situation that have serious implications for ML, and highlight the shortcomings of current approaches as far as learning agents are concerned.

2.1 Agents must learn very quickly

People will only interact with machines if they learn very quickly. Each example given requires significant attention from the user. It is essential that the agent begins to pull its weight by performing useful actions very soon. Only when it can be seen to be shouldering some of the user's burden will he or she be encouraged to invest the time and energy demanded by further instruction. Learning after one or two examples is desirable; half a dozen is already a large number for the user to provide unless he can see the agent volunteering some assistance.

From a conventional ML standpoint it is hard to imagine learning anything sensible from so few examples—especially when we consider the enormous size of the feature space. Nevertheless it must be done if learning agents are to find widespread practical application.

2.2 What's in an example?—the agent must decide

Standard ML assumes that each example is fully identified, carved out from the rest of the world, and distinguished from it. In the simplest case, an example comprises a set of identifiable feature values, along with a class. However, for a learning agent it is by no means clear what parts of the current environmental state constitute a given example. Typically there is a large number of features that might be assigned to the example; the question is, which ones are likely to be relevant? Users will not want to waste time answering such questions.

For instance, an example might be a piece of text pointed to on the screen—a word, say—and possible attributes might include the identity of the word, linguistic features (such as prefixes or suffixes), typographical properties (such as capitalization, font, style), positional attributes (such as position on the screen, on the line, in the paragraph), immediate contextual features (such as the preceding or following character, word, or words), and remote contextual features (such as the heading of the column it falls in, or the style of the paragraph it occupies).

The huge proliferation of features—there might be dozens, hundreds, or even thousands of potential ones—and the need to learn from a small number of examples—like two!—radically affects the nature of the ML problem. It becomes essential to focus the learner's attention on a particular subset of the feature space. Information that might be used for feature selection includes *a priori* domain knowledge, explicit instructions from the user, and the nature of already-learned rules. The learner must be capable of invoking focusing operations dynamically, to broaden or restrict the subset of features that are actually being considered depending on how learning is progressing.

Current focusing techniques do little more than switch features in or out according to the demands of the task. But a softer approach may be more productive, for example associating a weight with individual features and taking this into account during induction. The question then becomes how to shift the weight distribution as the focus of attention changes.

2.3 Sequential concepts are important

Tasks are generally executed as a sequential series of steps, and learning agents must be capable of inferring structure from observing the steps. Little work in ML has addressed inherently sequential concepts: the few exceptions (such as SPARC/E, Dietterich and Michalski, 1986; and TDAG, Laird and Saul, 1994) have for various reasons not been applied to PBD.

One useful sequential technique is the *k*-reversible grammatical inference method of Angluin (1982). Schlimmer and Hermens (1993) adopted a modified version of this algorithm to form a reversible automaton that captured the structure of several repetitive data entry tasks. Each data entry record was viewed as a sentence from a grammar in which words represented tokens. Not only did the system predict inputs, it also created customized data entry forms directly from the inferred automaton.

In this application successive sentences are clearly differentiated from one another. It is more common to have to deal with a single, continuous, unsegmented behavior sequence, and this renders standard grammatical inference techniques inapplicable. A further complication is the need for real-time learning: this means that the time taken by the inference algorithm may grow at most linearly with the number of elements in the sequence.

A linear-time algorithm for inferring hierarchical structure from sequences has recently been described (Nevill-Manning, 1995), which is based on an idea by Maulsby. However, its application to the actual construction of learning agents has yet to be demonstrated.

2.4 Learning must be sustainable

Interacting with learning agents is a truly incremental endeavor and learning must be sustained over significant periods of time. In contrast, most widely-used ML schemes are non-incremental in nature. For example, top-down decision tree learners (like C4.5, Quinlan, 1993) require all examples to be presented at once. Although incremental versions have been investigated (e.g. ID4, Schlimmer and Fischer, 1986; ID5R, Utgoff, 1989), they have not found much application: they either make serious sacrifices in inductive power or appear to have heavy resource requirements. Moreover, the problem of incremental learning is seriously exacerbated by the fact that the feature space is constantly shifting due to dynamic focusing of attention.

In practice, people frequently simulate incremental learning by re-applying a non-incremental induction algorithm from scratch at regular intervals. Unfortunately, sustained learning is impossible in principle unless it takes time that grows at most linearly with the number of examples seen, and constant re-learning from scratch violates this precept.

Mitchell's (1978) version space paradigm shows how to construct a learner which processes examples incrementally and yet remains insensitive to their order. However, although the concepts learned are independent of presentation order, the amount of search involved is not. In practice this still places a high premium on the user's ability to select a well-ordered sequence of examples.

Incremental learners have been investigated in the realm of inductive logic programming. MARVIN is an early example (Sammut and Banerji, 1986). The user begins by presenting an example of the desired concept, whereupon MARVIN proceeds to ask questions to eliminate possible hypotheses. While learning a concept, it modifies its current hypothesis by generalization and specialization transforms until it converges to the target concept. While this is an appealing model, in practice MARVIN asks a huge number of apparently trivial questions that would certainly not be tolerated by any serious user.

2.5 Learning must be reversible

One of the cardinal principles of HCI is that any action a user takes should be reversible: users should always be able to undo the effect of their work. This is no problem, of course, for non-incremental ML techniques: one merely erases the error from the database of examples before re-learning the set of rules. However, we argued above that incremental learning techniques are essential, and the question of how to reverse the effect of past erroneous input is an open challenge for ML.

Although an *undo* operation is sufficiently powerful to allow users to correct their errors in conventional computer dialogues, errors that occur when interacting with a learning agent are likely to lead to a breakdown of communication that requires more extensive effort to repair. The "breakdown-and-repair" process has been discussed extensively by Winograd and Flores (1986), who argue that it is difficult, if not impossible, for a computer system to identify breakdowns in communication because doing so requires stepping outside the range of inputs that were explicitly anticipated by its programmer. Certainly the problem of identifying breakdown in the learning process, and pinning it down to a specific erroneous input, presents a fascinating research challenge.

Statistical noise is not generally an issue with user-supplied examples. Two sources of apparent noise are user error, and an inadequate set of attributes, and these have already been discussed. Rather than treating noise as inherent and coping with it using probabilistic induction techniques, it is more profitable to isolate sources of apparent noise and rectify them individually.

3. THE ROLE OF INTERACTION

As we have seen, significant challenges are presented by a learning agent that is likely to be usable in practice—indeed, the problems are so severe that they cast doubt on the viability of the whole enterprise. Fortunately, many aspects of the interactive situation can be exploited to provide significant leverage for ML.

3.1 Learning agents can take initiative

Agents can take the initiative by proposing examples to the user: indeed, they are expected to do so, for it is often only through making suggestions that they actually provide assistance. Relatively little ML research has addressed active learning, which is surprising because learning systems have much to gain by showing initiative—in particular, by actively posing test examples rather than passively awaiting their appearance.

By taking the initiative, an agent can reduce free variation in the way the user performs a task. In a procedural setting, free variation in an action sequence is likely to confound attempts to learn. However, immediate input from the agent serves to guide the user into a consistent action sequence. By offering predictions as early as possible, an agent encourages consistency.

Great sensitivity is required when making suggestions to the user—we have already mentioned how irritating it is for MARVIN to ask hordes of trivial questions. In the user's mind, there is an important distinction between an agent performing some action on behalf of the user, and an agent asking for information or making a suggestion that is not directed towards easing the user's load. In the former case, it is apparent that the agent is "trying" to help—even if it is misguided—and is directing energy to the task at hand. In the latter, the agent seeks information for its own internal purposes and the user must take it on trust that it is worth responding.

3.2 Agents should say what they are doing

When taking action, agents should articulate the operations they are performing in terms that the user understands. This fulfills three important functions. First, it gives insight into the rules that have been learned for the task, thereby increasing the user's confidence in the actions. Second, conceptual errors on the learner's part may be revealed more quickly. Third, talking about the task helps to communicate the agent's conceptual model to the user, increasing the chance that he or she will in future adopt terms that the agent understands.

This has been observed in a study of user interaction with TURVY, a simulated agent (Maulsby *et al*., 1993). The task is a set of repetitive editing operations on a bibliographic database. Users began by offering instructions in bibliographic terms, such as "look for the last author's surname ...". TURVY, knowing nothing about bibliographic terminology, gave its verbal feedback in language like "I'm searching for the first colon following a full stop." In subsequent interactions, users quickly and

naturally abandoned their task-level terminology and adopted TURVY's lexicographically-oriented language; which meant that the hints they gave had a better chance of being understood.

3.3 Explaining the rules to a person is an unsolved problem

Communicating rules formed by a learning agent is problematic. The advantage of PBD is that users operate in a familiar environment, and communication occurs via objects with which they are familiar. Talking about the task requires the introduction of artifacts to represent generalized actions, shifting the communication from concrete to abstract. One technique for communicating generalized actions is to use a "storyboard" containing snapshots of a prototypical example before and after each action (Kurlander and Feiner, 1993; Modugno and Myers, 1993; Lieberman, 1993). Unfortunately, considerable cognitive load is involved in identifying the changes that have occurred between snapshots, which reduces the perspicacity of the representation.

Conveying the overall flow of control of the agent's program is even more problematic than representing a single action. Constructs such as loops and branches must be communicated in a way that draws upon concepts with which the user is familiar. Consider, for example, the procedural knowledge represented by the instructions for setting the time on a VCR—the widely recognized difficulty that people have with this task underlines the difficulty of communicating procedures even for skilled technical writers, let alone for automated agents. One obvious approach is to represent procedures as flowcharts composed of storyboard segments (Sassin and Bocionek, 1996); however, non-specialist users may be unable to comprehend a non-trivial flowchart even though its meaning is clear to a programmer.

3.4 Instructional hints can serve to alter the learner's bias

The dominant learning problem in intelligent agents is the determination of an appropriate subset of features to use. We have found that users readily and spontaneously give "hints" that provide information about which features are appropriate. If the interface provides appropriate communication channels, these hints can be used to hasten learning. They may be linguistic—users describe them in natural language—or deictic—users point to relevant parts of the screen. They can be used to rank features in relevance order. This enables the learner to infer the concept more rapidly by concentrating on particular features.

The problem is that hints are invariably ambiguous—even when they appear to be fairly well specified. We are developing an instructional model that involves a small number of basic instruction types, each involving a fixed set of parameters. In practice, instructions are only partially specified, in that some of the parameters are unbound. Sometimes this means that the hint translates into a *constraint* rather than a fully-specified instruction. Sometimes the agent must guess the unspecified parameters, bearing in mind that incorrect guesses may lead to a communication breakdown later. Sometimes the agent will verbalize the assumptions it is making, or even ask the user to resolve the ambiguity explicitly.

3.5 Learning can help disambiguate the instructor's hints

Linguistic hints pose a problem of interpretation. Processing verbal input apparently calls for full natural language recognition and understanding—a problem that is "AI-

complete". However, when hints are used merely to bias a learning algorithm, the requirement for full understanding can be relaxed.

Simple keyword spotting, augmented with thesaurus lookup, often provides enough information to focus a learner. For example, "look for an italicized word that begins with a capital letter" might yield *italic*, *word*, *begin*, *upper-case*, which would provide a starting point for focusing a learner (note the translation of "capital" to "upper-case", which is accomplished by a simple thesaurus lookup). Of course, the bias may need to be expanded, or contracted, in order to learn an acceptable representation of the concept—but such mechanisms are necessary anyway. What the hint does is provide a suitable, and possibly very useful, starting point.

Even when a keyword appears in a context that negates it—suppose the hint "please avoid words in italics" yielded merely the keywords *word*, *italic*—the agent can nevertheless learn the negation from examples of the concept rather than from the hint. The use of ambiguous hints as focusing instructions, combined with actual examples of the concept, yields a synergy: language enables learning, and learning disambiguates language.

CONCLUSIONS

Interactive environments pose interesting challenges and offer new sources of power to machine learners. To provide real assistance to users, learning agents must operate autonomously, accurately and unobtrusively. Learning must occur quickly, even when the problem is vastly underconstrained, and it can only do so by taking advantage of the assistance that users can provide. Agents must take initiative and make decisions about the scope of the learning problem. They must not only perform classification, but learn the sequential structure of the task. Finally, they should communicate with the user in a comfortable and comprehensible manner.

PBD presents opportunities for the ML community to devise new learning techniques, and for the HCI community to open up flexible communication channels between users and their agents.

REFERENCES

Angluin, D. (1982) "Inference of reversible languages." *J ACM 29*(3), pp. 741–765.

Cypher, A. (ed.) (1993) *Watch what I do: programming by demonstration.* MIT Press. Cambridge MA.

Dietterich, T.G. and Michalski, R.S. (1986) "Learning to predict sequences." In *Machine learning: an artificial intelligence approach II*, edited by R.S. Michalski, J.G. Carbonell and T.M. Mitchell, 63–106. Morgan Kaufmann, Los Altos, CA.

Kurlander, D. and Feiner, S. (1993) "A history-based macro by example system." In *Watch what I do: programming by demonstration.*, edited by A. Cypher, pp. 320–338. MIT Press. Cambridge MA.

Laird, R. and Saul, R. (1994) "Discrete sequence prediction and its applications." *Machine Learning 15*(1): 43–68; April.

Lieberman, H. (1993) "Mondrian: a teachable graphical editor." In *Watch what I do: programming by demonstration.*, edited by A. Cypher, pp. 340–357. MIT Press. Cambridge MA.

Maulsby, D., Greenberg, S. and Mander, R. (1993) "Prototyping an intelligent agent through Wizard of Oz," in *Proc InterCHI'93*, pp. 277–285. Amsterdam.

Maulsby, D. (1994) “Instructible agents.” PhD thesis, Department of Computer Science, University of Calgary.

Mitchell, T.M. (1978) “Version space: an approach to concept learning.” PhD thesis, Stanford University.

Modugno, F. and Myers, B.A. (1993) “Graphical representation and feedback in a PBD system.” In *Watch what I do: programming by demonstration.*, edited by A. Cypher, pp. 415–422. MIT Press. Cambridge MA.

Nevill-Manning, C.G. and Witten, I.H. (1995) “Detecting sequential structure.” *Proc Workshop on Programming by Demonstration*, ML 95, Tahoe City, CA, pp. 49–56.

Quinlan, J.R. (1993) *C4.5: programs for machine learning*. Morgan Kaufmann, San Mateo, CA.

Sammut, C. and Banerji, R. (1986) “Learning concepts by asking questions.” In *Machine Learning Vol II*, edited by R.S. Michalski, J.G. Carbonell and T.M. Mitchell, pp. 167–191. Morgan Kaufmann.

Sassin, M. and Bocionek, S. (1996) “Meeting the user’s intention in programming by demonstration systems.” *Proc Acquisition Learning and Demonstration Workshop*, AAAI Spring Symposium, Stanford, CA, pp. 131–135.

Schlimmer, J.C. and Fischer, D. (1986) “A case study of incremental concept induction.” *Proc Fifth Annual Conference on AI*, Philadelphia, PA, pp. 496–501.

Schlimmer, J.C. and Hermens, L.A. (1993) “Software agents: completing patterns and constructing user interfaces”, *Journal of Artificial Intelligence Research*, 1, pp. 61-89.

Utgoff, P.E. (1989) “Incremental induction of decision trees.” *Machine Learning 4*(2), pp. 161–186.

Winograd, T. and Flores, F. (1986) *Understanding computers and cognition.* Ablex, Norwood, NJ.