Combining learning from instruction with recovery from incorrect knowledge

Douglas J. Pearson
Artificial Intelligence Laboratory
The University of Michigan
1101 Beal Ave.
Ann Arbor, MI 48109
dpearson@umich.edu

Scott B. Huffman
Price Waterhouse Technology Center
68 Willow Road
Menlo Park, CA 94025
huffman@tc.pw.com

1 Introduction

Autonomous learning agents acquire knowledge from a variety of sources over their lifetimes. Some knowledge is given to the agent "at birth"; the rest is learned, through methods like autonomous exploration/observation of the environment, interactive dialogues with an instructor, etc. Unfortunately, there is no guarantee that any of this knowledge is correct. Thus, a learning agent needs the ability to recover from incorrect knowledge when that knowledge leads to performance failures.

Recovering from incorrect knowledge that has caused a performance failure is difficult for two reasons. First, the agent must discover how to remedy the performance failure – how to get out of a bad situation and back on the path towards its goals. Second, the agent must determine what knowledge caused the failure (a difficult credit assignment problem) and how to correct that knowledge. Thus, techniques for recovery from incorrect knowledge generally rely on weak inductive methods to correct the knowledge and then re-planning to resolve the current performance failure (e.g. EITHER [Ourston and Mooney, 1990], CLIPS-R [Murphy and Pazzani, 1994] and Classifiers [Holland, 1986]). IMPROV [Pearson and Laird, 1995], the recovery system used in this work, uses a weakly constrained search in the environment to find a recovery path, and an inductive category learner for credit assignment and knowledge correction.

In this note, we describe an extension to IMPROV that allows an instructor to interactively guide various stages of the recovery process. The extension integrates IMPROV with Instructo-Soar [Huffman and Laird, 1994; Huffman, 1994], an instructable agent that engages in an interactive natural language dialogue with an instructor to learn new tasks and other kinds of domain knowledge.

Combining learning from instruction with recovery from incorrect knowledge makes both techniques stronger. Instruction aids recovery by guiding the recovery process, producing faster and more accurate recovery and knowledge correction. Recovery aids instruction by providing a way to deal with incorrect knowledge learned from either bad instructions or incorrect generalizations of good instructions. This allows an instructable agent to avoid being either overconfident (i.e., to assume everything it learns is correct) or paranoid (i.e., to ask the instructor to verify everything it learns, because it might be wrong). Rather, the agent can assume the instructions it receives are mostly correct, and can make generalizations during learning without continually asking for verification, because it will be able to recover if performance errors arise later.

In what follows, we use a running example to describe the integrated instruction/recovery agent's performance. First, we describe an instruction scenario that introduces errors into the agent's knowledge. Next, we discuss IMPROV's recovery technique applied to the error, first with no instruction available, and then with instruction for each of the stages of the recovery process in turn.

2 Errorful knowledge learned from instruction

Instructo-Soar uses a technique called *situated explanation* [Huffman, 1994; Huffman and Laird, 1994] to learn from instructions. Situated explanation combines analytic and inductive learning methods to learn new procedures and extensions of procedures for novel situations, and other domain knowledge such as control knowledge, knowledge of operators' effects, and state inferences.

For each instruction, situated explanation involves first determining what situation (state and goal) the instruction applies to – either the current situation, or a hypothetical one specified in the language of the instruction. Then, the agent attempts to explain, using a forward projection, why the instruction will lead to success in the situation. If this explanation succeeds, the agent can learn general knowledge from the instruction (as in standard EBL [Mitchell *et al.*, 1986]). If the explanation fails, it means the agent is missing some knowledge required to complete the explanation. The missing knowledge can be acquired either through further instruction, or in some cases through simple induction over the "gap" in the incomplete explanation. However, either instructions or the agent's inductions can be incorrect and lead to learning errorful knowledge.

Instructo-Soar's domain includes a table with red and green buttons and a light on it (see Figure 1 (a)). The red button turns the light on and off, but the agent does not know this. Consider an example in which the agent is first taught a general procedure for how to push buttons, using the instructions in Figure 1 (b). Some time later, the instructor says:

"To turn on the light, push the red button."

To perform a situated explanation, the agent first determines that this instruction applies to a situation where the goal is to turn on the light. Then, it forward projects the action of pushing the red button in that situation. However, since it does not know that the button affects the light, this projection does not explain how pushing the button causes the goal of turning on the light to be reached.

In this case, the agent makes a simple inductive leap to complete the explanation. It guesses that since there is a single action specified to reach a goal, that action directly causes the goal to be reached – e.g., pushing the button has the effect of turning on the light. Its inductive bias specifies that in this type of induction, the types of objects involved and their relationship to one another are important (e.g. that the button is a button and the light is a light, and that they are on the same table) but other features, like the button's color, are not. Thus, the agent learns a rule that says "pushing any button on the same table as a light causes the light to turn on." This rule allows the agent to complete its explanation of the original instruction, producing a control rule that says "if the goal is to turn on a light on a table, choose the action of pushing some button on that table."

Previous versions of Instructo-Soar would ask the instructor to verify the inductive leap – Instructo-Soar was a "paranoid" agent, afraid to learn any wrong knowledge. For these experiments, we turned off the verification step. Instructo-Soar simply makes its induction (which in this case is overgeneral) and carries on. If errors in performance arise later due to the induced knowledge, the agent can use recovery techniques to correct the knowledge.

3 IMPROV's recovery technique

We will first describe each stage of IMPROV's recovery process and then how it can be informed by instruction. There are three principle stages to error recovery: (1) Detecting an error, (2) Finding a plan that, when executed, leads from the errorful state to achievement of the current goal, and (3) Identifying the piece of knowledge which led to the error and correcting it. IMPROV achieves these stages by (1) recognizing when the agent is no longer making progress in its problem solving, (2) using previous case knowledge to guide a search for a successful plan, and (3) comparing successful and incorrect plans to identify the incorrect operators and then training an inductive learner on the results of executing the plans, which in turn leads to new case knowledge.

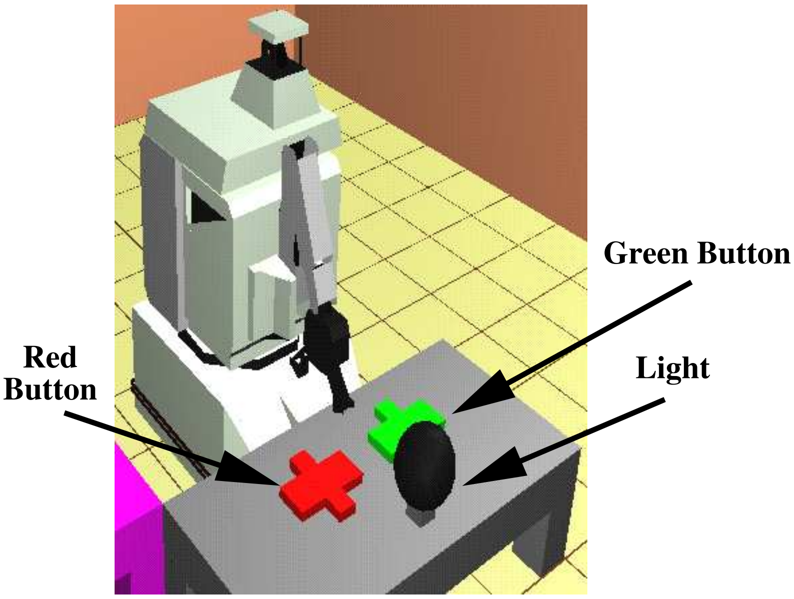
In our example, when the agent is asked to turn on the light, its overgeneral knowledge ("pushing any button causes the light to turn on") may lead it to push the wrong (green) button. When it does, IMPROV detects the failure (the light doesn't come on), searches for the correct plan (pushing the red button) and then learns that to turn on the light, the agent must push the red button, not the green one. Alternatively, at any (or all) stages of the recovery process, the instructor can provide simple guidance that speeds up the process by avoiding search.

3.1 Error Detection

IMPROV detects execution errors by recognizing when the agent is unable to make progress towards its goals. This is done by detecting when no operator is proposed for the current goal. In our example, IMPROV detects an error when the agent does not know what to do after pushing the wrong button and the light doesn't come on. This method is augmented with loop detection, where the agent recognizes that it has returned to an earlier problem state, which is another indication that an error has occurred. For example, if the agent had already pushed the green button and proposed an operator to push it again, this would be detected as an error.

3.2 Finding a plan that reaches the goal

After detecting an error, the agent must find a way round that error and reach its goal. IMPROV searches for alternative plans, executing each in turn, to find one that satisfies the current goal. This search is biased towards plans which are believed to be most likely to succeed, based on previously seen training cases. If additional causal or diagnostic knowledge is available to the agent, this search can be further focused; however, we do not assume the presence of such knowledge. In our example, if the agent has seen instances of pushing a button and having a light come on, then push-



"Push the green button."

How do I do that?

"Move to the grey table."
"Move your hand above the green button."
"Move down."
"The operator is finished."

Figure 1: (a) The example domain and (b) Instructions on how to push a button

button is associated with turning on lights. As a result, when the error occurs, plans that include the push-button operator are preferred over plans that do not include operators associated with turning on lights. If the agent has no previous case knowledge, this search defaults to iterative deepening.

3.3 Identifying the cause of the error and learning a correction

There are two credit assignment problems to be solved in identifying and correcting an error in the agent's knowledge: first, identifying which operator is incorrect and second, identifying how that operator's knowledge is incorrect.

IMPROV identifies which operator(s) have incorrect knowledge by comparing the successful plan to the original (incorrect) plan. Ordering and instantiation differences between the plans indicate potentially incorrect operators. In the light example, both the correct and incorrect plans contain a single operator (push-button) with different instantiations, so it is identified as the erroneous operator.

Once the operators with incorrect knowledge have been identified, IMPROV uses an incremental inductive category learner, SCA [Miller, 1991; Miller, 1993], to identify how the operator is incorrect. The category being learned is which operator to select for a given state and goal. IMPROV is currently limited to correcting operator precondition knowledge. To avoid the error in future, the agent must decide which features of the state or goal caused the operator to succeed or fail. In our example, the agent must learn that the reason pushing the red button works is because the button is red, not because it is the button on the right, or because the robot's hand is open or closed etc.[1]

In its weakest form, IMPROV simply relies on the induction made by SCA to determine the features which are responsible for the failure, and revises its domain knowledge accordingly. As this induction is made without access to a causal theory or other knowledge source, the inductive guess may be wrong, leading to a future error and the need for another recovery. However, as the planner is complete and SCA can represent arbitrarily complex categories [Miller, 1993],[2] the system will eventually converge to the correct knowledge.

4 Instruction to inform recovery

We have augmented IMPROV by allowing an instructor to interrupt the recovery process at any time and provide instructions that guide the recovery. Figure 2 lists how instructions can apply to each phase of recovery. In particular, instructions can be used to warn the agent that an error is about to happen, to help the agent find the correct way to achieve its current goal or to help identify the reasons for the error.

4.1 Error detection

IMPROV's instructor can indicate that an error is about to happen by interrupting the agent with the command "Stop!" Our agent assumes that this command means that the currently selected operator will lead to a execution error, rather than to the current goal. In our example, if the agent chooses the wrong

[1] This explanation is limited by the agent's representation language. To determine the real cause, that the red button is electrically connected to the light, the agent would need a deeper theory.

[2] Assuming the system's representation is sufficient to represent the correct operator preconditions, that actions do not destructively modify the environment and that a solution path exists.

	Weak Recovery Method	Assistance from Instructor	Example Instructions
Error Detection	Detect inability to make progress toward the goal.	Instructor indicates a failure is about to occur.	"Stop!"
Finding the correction path	Case−guided search	Instruction−guided search	"Push the red button." "Think about the red button."
Identifying the cause of the failure.	Induction to identify differences between instances.	Instructor indicates important differences between instances.	"Think about color."

Figure 2: The stages of a recovery and how instruction can help.

(green) button, the instructor may say "Stop", realizing an error is about to occur as the agent moves it's hand over the wrong button. The system then treats this situation as if an error had occurred and begins the recovery process. IMPROV records the operator push-button(green), along with the state when the operator was chosen, as a negative training instance for the inductive module and starts a search for the correct way to turn on the light.

4.2 Finding a plan that reaches the goal

Once an error has been detected, IMPROV searches for a correct plan to achieve the current goal. At any time during this search, the instructor may provide guidance in the form of one or more steps in that plan. In our example, the instructor can interrupt the search by saying:

"Push the red button."

This leads the agent to prefer plans that include pushing the red button. As the agent believes pushing any button turns on the light, it believes this plan will succeed and immediately executes it. After it succeeds, the agent's knowledge is corrected just as if it had discovered the plan through search instead of instruction.

Because human instructors often give incomplete instructions, IMPROV can also use incomplete specifications such as:

"Think about the red button."

This instruction directs IMPROV to prefer plans that involve the red button over plans that don't. Depending on the previous case knowledge, IMPROV might still try executing some other operation on the red button before "push-button(red)". However it would prefer anything involving the red button over "push-button(green)" because of the instructor's guidance.

The instructions are used as search control. In the event that the instructions are incorrect, IMPROV will try the suggested path, see that it fails and continue searching for a correct plan.

4.3 Identifying the cause of the error and learning a correction

Once a correct plan has been found (either through search or with the aid of instruction), IMPROV needs to identify the cause of the error. IMPROV must determine the key difference(s) between the plan that succeeded and the plan that failed. Identifying which differences are reasons for the failure is a difficult credit assignment problem. In our example, the green button is to the right of the red button. Without help from the instructor, IMPROV must guess whether the reason push-button(green) failed was because the button was on the right, or because it was green. If it makes the wrong induction, the agent may fail later (e.g. if it ever encounters a pair of buttons arranged the other way round). However, the instructor can interrupt the learning process and guide the choice of features for the induction. In our example, if the instructor says:

"Think about color."

this leads IMPROV to focus on the colors of the buttons (rather than their positions) and ensures that the correct induction is made. In this case, learning that to turn on the light, the push-button operator should not be selected for a green button but should be selected for a red button.

5 Conclusion

We have demonstrated how recovery aids instruction, by providing a method for handling incorrect instructions, or incorrect inductions from instructions. We have also demonstrated how instruction can aid recovery, by avoiding search and guiding the induction process. Our example and the instructions used are preliminary and fairly contrived, but they illustrate how by integrating a weak, but robust, error recovery strategy with a strong, but potentially brittle, instruction strategy both are improved.

References

[Holland, 1986] John H. Holland. Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An artificial intelligence approach, Volume II.* Morgan Kaufmann, 1986.

[Huffman and Laird, 1994] S. B. Huffman and J. E. Laird. Learning from highly flexible tutorial instruction. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*, Seattle, WA, 1994.

[Huffman, 1994] S. B. Huffman. *Instructable autonomous agents.* PhD thesis, University of Michigan, Dept. of Electrical Engineering and Computer Science, 1994.

[Miller, 1991] Craig M. Miller. A constraint-motivated model of concept formation. In *The Thirteenth Annual Conference of the Cognitive Science Society*, pages 827–831, 1991.

[Miller, 1993] Craig M. Miller. *A model of concept acquisition in the context of a unified theory of cognition.* PhD thesis, The University of Michigan, Dept. of Computer Science and Electrical Engineering, 1993.

[Mitchell *et al.*, 1986] Tom M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1, 1986.

[Murphy and Pazzani, 1994] Patrick M. Murphy and Michael J. Pazzani. Revision of production system rule-bases. In *Proceedings of the International Conference on Machine Learning*, pages 199–207, 1994.

[Ourston and Mooney, 1990] Dirk Ourston and Raymond J. Mooney. Changing the rules: A comprehensive approach to theory refinement. In *Proceedings of the National Conference on Artificial Intelligence*, pages 815–820, 1990.

[Pearson and Laird, 1995] Douglas J. Pearson and John E. Laird. Toward incremental knowledge correction for agents in complex environments. In Stephen Muggleton, Donald Michie, and Koichi Furukawa, editors, *Machine Intelligence*, volume 15. Oxford University Press, 1995.