ABSTRACT

This paper describes a substructure discovery system, PLAND, that combines empirical learning methods with knowledge intensive learning algorithms. Unlike other systems which combine similarity-difference-based and explanation-based learning techniques at a single level, the PLAND system uses knowledge to direct the learning process on three distinct levels. This multi-leveled approach to learning allows a system to be more flexible and adaptive to the current learning task than with a single level approach.

1 Introduction

There have been two major emphases in machine learning, similarity-difference-based learning (SDBL) [4, 5, 17, 19] and explanation-based learning (EBL) [2, 9]. Each method of learning has its strengths but also weaknesses. Recently, integrated systems have been built that combine these methods to obtain the benefits of each while reducing their respective disadvantages. Although the systems combining SDBL and EBL have been successful, they use the two types of algorithms on a single level of abstraction. This paper describes an approach used by the PLAND system that uses a synthesis of these methods on multiple levels of learning. The PLAND integrated learning system uses background knowledge on three different levels to guide its learning task. Using three levels is a conceptually clean way to approach the task of discovering macro-operators, the goal of the PLAND system.

Integrated learning systems combine SDBL and EBL to overcome the weaknesses of each system. EBL is most useful when the system has complete domain knowledge about the given task. With EBL, a system can deduce a verifiable, generalized rule from a single example. Unfortunately, EBL does not handle situations when there is not complete, tractable domain knowledge. In addition, these systems do not learn at the knowledge level [3], the

2 Combining Learning Methods

This section discusses four systems that use a single level approach to the combination of SDBL and EBL methods, representing other current research. There are two ways that these systems combine learning methods. One way is for a SDBL algorithm to call an EBL algorithm to explain the empirical results. This produces an explanation structure for the proposed solution. This structure can be used in the refinement of the discovered rule. In the other approach, an EBL algorithm calls a SDBL algorithm to find/propose some knowledge that is missing in the domain theory. For example, when a branch of the proof rules learned improve performance only. SDBL finds common attributes among given examples to induce new descriptive rules for classifying the events. It does not require complete domain knowledge, in fact few systems use any sort of background knowledge. This is the weakness of SDBL. While it can always determine an empirically valid generalized description, the results may be whimsical with no domain related reason for using the selected attributes and values. The goal of integrated learning is to combine these methods in a fashion that allows the goal directedness of EBL to help guide the selection of features by an SDBL system and to allow an SDBL system to provide needed knowledge to the incomplete domain theory of an EBL system. An SDBL system can, in addition, be used to determine empirically valid examples to pass to an EBL system. It would be intractable for an EBL system to discover these examples by itself. A multi-levelled approach to integrating SDBL and EBL, as demonstrated by the PLAND system, is advantageous because it allows more interaction between the algorithms. This interaction provides a mechanism for the learning system to focus on the goal. In the next section a brief description of some current research in combining empirical and knowledge intensive learning algorithms is presented. This is followed by a discussion of why a multi-levelled synthesis of these learning methods is advantageous. The fourth section presents a brief description of PLAND, a substructure discovery system for finding macro-operators (macros) in observed action sequences. That section explains how PLAND uses knowledge at three different levels to guide the discovery process. The fifth section contains an example run of PLAND. This is followed by a discussion of possible future work and concluding remarks.

This research was partially supported by the Defense Advanced Research Projects Agency under grant N00014-87-K-0874 and by a gift from Texas Instruments, Inc.

**"Similarity-difference-based learning is the same as what was previously referred to as similarity-based learning (SBL) or empirical learning. However, these systems do not learn by observing only similarities. The differences between events also play an important role."**

ABSTRACT

This paper describes a substructure discovery system, PLAND, that combines empirical learning methods with knowledge intensive learning algorithms. Unlike other systems which combine similarity-difference-based and explanation-based learning techniques at a single level, the PLAND system uses knowledge to direct the learning process on three distinct levels. This multi-leveled approach to learning allows a system to be more flexible and adaptive to the current learning task than with a single level approach.

1 Introduction

There have been two major emphases in machine learning, similarity-difference-based learning (SDBL) [4, 5, 17, 19] and explanation-based learning (EBL) [2, 9]. Each method of learning has its strengths but also weaknesses. Recently, integrated systems have been built that combine these methods to obtain the benefits of each while reducing their respective disadvantages. Although the systems combining SDBL and EBL have been successful, they use the two types of algorithms on a single level of abstraction. This paper describes an approach used by the PLAND system that uses a synthesis of these methods on multiple levels of learning. The PLAND integrated learning system uses background knowledge on three different levels to guide its learning task. Using three levels is a conceptually clean way to approach the task of discovering macro-operators, the goal of the PLAND system.

Integrated learning systems combine SDBL and EBL to overcome the weaknesses of each system. EBL is most useful when the system has complete domain knowledge about the given task. With EBL, a system can deduce a verifiable, generalized rule from a single example. Unfortunately, EBL does not handle situations when there is not complete, tractable domain knowledge. In addition, these systems do not learn at the knowledge level [3], the

2 Combining Learning Methods

This section discusses four systems that use a single level approach to the combination of SDBL and EBL methods, representing other current research. There are two ways that these systems combine learning methods. One way is for a SDBL algorithm to call an EBL algorithm to explain the empirical results. This produces an explanation structure for the proposed solution. This structure can be used in the refinement of the discovered rule. In the other approach, an EBL algorithm calls a SDBL algorithm to find/propose some knowledge that is missing in the domain theory. For example, when a branch of the proof rules learned improve performance only. SDBL finds common attributes among given examples to induce new descriptive rules for classifying the events. It does not require complete domain knowledge, in fact few systems use any sort of background knowledge. This is the weakness of SDBL. While it can always determine an empirically valid generalized description, the results may be whimsical with no domain related reason for using the selected attributes and values. The goal of integrated learning is to combine these methods in a fashion that allows the goal directedness of EBL to help guide the selection of features by an SDBL system and to allow an SDBL system to provide needed knowledge to the incomplete domain theory of an EBL system. An SDBL system can, in addition, be used to determine empirically valid examples to pass to an EBL system. It would be intractable for an EBL system to discover these examples by itself. A multi-levelled approach to integrating SDBL and EBL, as demonstrated by the PLAND system, is advantageous because it allows more interaction between the algorithms. This interaction provides a mechanism for the learning system to focus on the goal. In the next section a brief description of some current research in combining empirical and knowledge intensive learning algorithms is presented. This is followed by a discussion of why a multi-levelled synthesis of these learning methods is advantageous. The fourth section presents a brief description of PLAND, a substructure discovery system for finding macro-operators (macros) in observed action sequences. That section explains how PLAND uses knowledge at three different levels to guide the discovery process. The fifth section contains an example run of PLAND. This is followed by a discussion of possible future work and concluding remarks.

**"Similarity-difference-based learning is the same as what was previously referred to as similarity-based learning (SBL) or empirical learning. However, these systems do not learn by observing only similarities. The differences between events also play an important role."**
tree cannot be grounded with the current facts of the example or knowledge. Lebowitz [8] takes the first approach to improve the performance of UNIMEM [7]. This system uses the rules discovered by the SDBL portion of the algorithm as examples to be explained and further generalized by the EBL portion. By giving the generalizations discovered by the SDBL system as an EBL example, the EBL system is working with only empirically valid descriptions and not attempting to work on an intractable number of possibilities.

Using an approach similar to UNIMEM's, the CLUSTER/CA system [10] extends the SDBL system CLUSTER/S [17] by using a Goal Dependency Network (GDN) [18] to direct the building of classifications. The GDN indicates the goals of the current use of the learning algorithm and allows the system to build clusters that more closely match the desires of the user. The GDN can be thought of as a non-generalizing EBL system, i.e., a simple backward chaining mechanism. PLAND integrates SDBL and EBL in the same manner but uses multiple levels of knowledge instead of a single level.

In contrast, Pazzani has integrated SDBL and EBL methods by making the EBL component guide the process. OCCAM [11, 12] is an EBL system that learns to predict and explain the outcome of observed events. OCCAM uses similarities to build a causal explanation when the EBL portion does not contain complete background knowledge [13, 14]. In this implementation the EBL portion focuses the empirical method on aspects that directly relate to the given situation, thus reducing the attributes given to the SDBL system to those that are most significant to the problem.

Danyluk's system [1] uses a simple explanation of a terrorist event to guide the similarity-based frame matching algorithm to build a composite event of the situation. An important feature of this system is that EBL and SDBL methods can call each other recursively to direct the search for knowledge in both directions.

All the previous systems use knowledge at a single level of abstraction. PLAND goes beyond that by using background knowledge at three distinct levels. These levels provide more control during the search for new macro operators by allowing the system to focus on relevant portions of the input sequence and manipulating sequences of differing generalities by switching contexts.

3 Multi-level Integration

As demonstrated by the previous section, the current systems combine SDBL and EBL methods only on a single level. Clearly, additional knowledge can help a system to improve its performance. But various types of knowledge are applicable to different levels in the learning process. When control of the system is directed by a learning goal, the system needs the ability to understand what each level is accomplishing and the insight to determine if the current effort of a level is contributing to the learning goal or not. If the learning goals are well specified, then specific background knowledge can be accessed to direct the system in the proper way. By using background knowledge at different levels a system can focus on the learning task more expediently. Different levels allow the system to direct its search at a number of grain sizes. The highest (coarsest) level gives a general direction through the search space. Lower levels contribute by specifying what is the best move at the given level of abstraction. A level can look at what is being accomplished in the levels below it and decide when a shift in bias or search space would be useful. The next section describes the PLAND system and the following section describes how PLAND uses background knowledge at different levels to guide the learning process.

4 PLAND

The PLAND (PLAN Discovery) system [21] is a substructure discovery system for determining macro-operators (macros) that could be used in plans to carry out an observed sequence of action steps. Substructure discovery is the grouping of entities from an input event in such a manner that using the substructure as an atomic unit is cognitively more efficient than using the separate pieces. In other words, it is easier to reason about a set of smaller parts and their interrelations as a single object (substructure) than to continuously consider each part individually. A substructure is represented by a collection of nodes (objects) and relations between those objects. In the PLAND system only one type of relation is present, the temporal "follows" relation. This is the one relationship needed to represent that only a single action may follow or proceed another observed action in the system. This restriction does not apply to substructure discovery systems in general [6].

From a sequence of actions given as input, the PLAND system discovers three types of macro substructures: sequences, loops, and conditionals. A sequence macro is a block of actions that have been used in many places in the given action trace. Loops are defined as sequences that appear juxtaposed for at least a minimum number of iterations. PLAND finds repetitive sequences but does not learn the stopping conditions for these constructs. It assumes the loop body can be performed any number of iterations. In order to determine the stopping criteria for a loop, information about the change in the current state as a result of the execution of an action would have to be maintained. A system such as BAGGER [15, 16] has such knowledge about the effects of actions and analyses the states to determine the stopping criteria. Conditionals allow for a choice of actions at some particular point in time. Given the observed action trace $A(B+D)C^*$, an example of a discoverable conditional is $(A(B+D)C)^*$1. By nesting previously discovered macros within other macros the system is able to discover complex relationships between different macros and build up a hierarchical structure of the observed sequence of actions.

5 Use of Background Knowledge

In the current version of PLAND, background knowledge is expressed by rules in Horn clause form. The system does backward chaining through the rules to obtain an answer to a query about what to do at a certain point in the processing. PLAND uses background knowledge provided by the rules in three distinct

---

1As in formal grammar syntax, Kleene closure is represented by *.
ways. Figure 1 gives a schematic view of the PLAND algorithm. The components of the algorithm will be discussed in the remainder of this section. At the highest level, the background knowledge (BK) performs the function of meta-knowledge, determining which context should be used. The second level determines which agendas within a context may be processed. The lowest level consults potentially domain specific background knowledge to determine which discovered macros are useful and should be allowed in the generalization of future contexts.

The system loops until domain knowledge fails to indicate anything useful to determine, i.e., until it fails to return a context (line 1). Abstractly, a context represents the biases and concept space used in the search for particular macros. This highest level also determines which macros from a previous context are used in building a new, more general context. Within each context, agendas are used to determine where to search for macros. Because they are limited to a specific context, agendas define a smaller area of the search space. Knowledge is used to indicate when an agenda should be performed. After a macro is discovered, domain knowledge is again consulted to determine the new macro's usefulness. Usefulness depends upon the number of occurrences of the discovered operation and the goals it could help achieve. Before the macro is added to the context, the system again checks to ensure that the macro is as widely used as first anticipated. Going back to the top of the loop, the new knowledge is used in conjunction with any new macros discovered to determine if a new context should be constructed or a different, previously constructed context should be selected.

5.1 High Level Background Knowledge

At the highest level, the background knowledge rules control the levels of generalization performed on the action sequence. All the information about a given level of abstraction and how to process the set of observed actions are kept in a structure called a context. A context includes the input sequence of actions, the previously discovered macros, the agendas (explained later), and information about how macros overlap and subsume each other. The system can proceed to a more abstract level by creating a new context in which some of the actions have been generalized. The flexibility of this scheme is useful if the system is unsure about which level of generalization is appropriate for the problem. The system can work on one context for awhile, then swap contexts and work on a different one. The observed actions can be generalized by replacing groups of actions with macros or by generalizing specific portions of an action through the use of a fuzzy matching algorithm.

The fuzzy matching algorithm allows flexibility in determining which actions are considered equal. The system determines if a sequence is repeating by the "equality" test among actions. The fuzzy matcher can work at multiple levels of match such as forcing actions to be identically equal ("eq equal"), ignoring certain parts of the action, considering all actions equal, or performing user supplied LISP functions that determine equivalence (such as determining if an argument to an action is a member of an important set of objects). When it is determined that a new level of generalization is required, the matcher is run on the actions to produce more general actions forming the basis of a new context. Macros that replace actions in the original sequence are treated as actions in the new context. Thus there is no difference of application between discovered macros and primitive a priori given actions.

After searching for a macro, the background knowledge is inspected to see if the current context is still preferred over another. If a different context is wanted, the knowledge base returns the new context to be used. At this level the knowledge is used to control the level of generalization of the action steps. Thus the system can process the input at a higher level of abstraction after some macros have been found. If the search at this more generalized level is fruitless then the system can return to the lower, more detailed level to attempt to discover more useful macros for further generalizations. The switching between contexts allows the system to search for macros in differing description spaces.

This is a powerful mechanism because it allows the system to pursue many possible goals. If the system discovers a macro that indicates that the actions occurred in a certain environment, then it can create a context that generalizes some of these actions to help confirm that notion. For example, if the system discovers at the base level a "picking up cans" macro, it might try to confirm that the current environment is a grocery store. This could be accomplished by searching for a macro that pushes a shopping cart, and generalizing cans, bottles, boxes, and bags to "things in a grocery store" in a new context. After searching in this generalized context for grocery store macros without success, the system can return to the original context to look for more macros. Working at the primitive level it then might discover a macro for "pulling weeds", so using the knowledge of picking up cans and pulling weeds in combination one suspects the task is cleaning up a yard or a roadside. There may be many competing contexts in the system, though only one has control at a time. Consultation with the background knowledge in this fashion allows the system to discover macros that would be impossible to discover otherwise. A system without domain specific knowledge could not make logical guesses about which of the many possible generalizations has meaning to the problem at hand. The SDBL

1) Do while BK suggests a context to work in
2) Pop an agenda for selected context
3) If BK determines agenda is potentially productive
4) Execute agenda
5) If a macro is discovered
6) If macro meets BK criteria
7) Find all occurrences in context trace of discovered macro
8) If BK suggests macro still useful
9) Define macro within the context
10) Create agendas that use the new macro
11) Endif
12) Endif
13) Merge created agendas into stack for context
14) Endif
15) Endif
16) Enddo

Figure 1: PLAND Algorithm
mechanism that searches for the macros is being directed by the EBL component used at the context level.

5.2 Medium Level Background Knowledge
The background knowledge used at the second level helps direct the searching process for the macros through agenda control. An agenda indicates where to look for new macros in the given action trace for a context. There are many agendas competing for processor time, and a simple agenda control system manages their priorities. An agenda contains information on the position in the action sequence where the search for a macro is to begin. The previously found macros (within the current context) that can be used in building up a new macro are specified in the agenda. The type of macro to be searched for is indicated as being either a loop or a conditional. There is no explicit search for sequence macros, they are found as a byproduct of discovering other macro types.

Before any agenda is given control, the background knowledge is consulted to approve its applicability. Agendas could be checked for applicability when they are added to the stack but this would result in wasted effort since many agendas are never run or are changed when new macros are discovered. Currently, rules are fired only once to approve each agenda where otherwise they would be used many times. An agenda usually is approved and executed. If the knowledge indicates that searching is not to be done beyond a certain point in the input sequence, then those agendas are pruned when they have done sufficient search. This allows the background knowledge to focus the discovery process into areas of the action trace that are considered most likely to produce results. Information contained in the agenda can signify when the agenda should not be performed. This allows simple biases to be applied to macro learning. The knowledge used in this method can save substantial amounts of processing and significantly prune the search tree. Again, the EBL processes are guiding the SDBL system to look in promising areas since there is a large number of areas to explore. An EBL system could not find these macros on its own because there is not a clear, single example demonstrating the use of the macro. Rather, there are numerous occurrences of the macro and complete knowledge of its effects are not known at this time.

5.3 Low Level Background Knowledge
At the lowest level, background knowledge is used to control the macros allowed by the system. After finding the sequence of actions for a macro-operator, a query is posed to determine if the sequence meets any domain specific criteria expressed for macros. With simple rules for checking macro sequences the system is able to eliminate the generation of useless macros. This saves not only storage, but also the expense of handling agendas for the new unproductive macros and the searches they generate. After all occurrences of a macro have been identified, a second reference to background knowledge is made. This time the information is used to confirm the validity of the macro. With the information of exactly where the macro occurs, the usefulness of the macro can be better evaluated and if not as great as first suspected, the macro is eliminated.

Determined which hypothetical macros found in a context are likely to be useful depends upon given knowledge. There are many hypothetical macros generated; any sequence of actions that occurs more than twice in the input is a potentially useful sequence. These macros are labelled "partials" until the system determines that they are useful. Although partial macros are not used in the normal discovery process, all such partials must be kept because they are potentially useful and may become more important when used in conjunction with subsequently discovered macros. However, to convert all those sequences to macros would bring the system to a halt in its search for other useful macros, therefore partials are only made into macros when they meet domain specific criteria. Most SDBL systems would either ignore all the partials or promote all of them. By incorporating knowledge at this level, PLAND can promote only those partial macros having some great number of occurrences or that accomplish some specified task.

Knowledge at the lowest level can rid the system of macros having low utility. Although this seems trivial, this level of control determines whether a system finds a solution or runs out of space and/or time. Control like this is missing in many SDBL systems. Those systems can only make guesses at what is useful, such as this system does when no background knowledge is present. The implication that nothing valid can be done without background knowledge is not intended. In fact this system can find some useful results even when no knowledge is given.

6 Example
An example of how PLAND uses background knowledge is presented. PLAND is a domain independent system that can work with or without background knowledge. When no domain knowledge is given the system performs like a finite state machine constructor. More interesting results are achieved when the system has knowledge available to it. For this example, the system is given a trace of actions performed by a robot moving boxes from one room to an adjacent room. Figure 2 contains a map of two rooms showing the robot (circle), boxes, and the room layout. The robot moves the boxes from the room in which they reside to an identical position in the second room. The goal is to discover useful macros found from this trace for moving boxes.
from one room to the other. This requires the use of background knowledge. There are two important pieces of information given to the system. First, it is told that there is a doorway between positions (10,5) and (11,5). Second, knowledge indicates that if a doorway is used during the action performed, the robot is moving between two rooms. When more than one room is involved the system should create a new generalised context. In the new context the y coordinate of MOVE is ignored and the object of GRASP or UNGRASP is ignored. These items are generalised to facilitate the search for macros. The output from the program on this example is given in figure 3.

The observed action trace at the top of figure 3 is the input to the system. The move command has the syntax (MOVE to-x-position to-y-position), grasp has syntax (GRASP item), and ungrasp has syntax (UNGRASP item). In the first cycle, the system discovers two partial macros that the domain knowledge indicates should be made into macros, because the goal is to learn macros about moving between rooms and those that involve a door. The actual rules are not shown because of the complexities of the rules for promoting a macro are described in figure 4*.

These two macros are for moving through the doorway (one from left to right, M:1.02; one from right to left, M:1.01).

At this point the knowledge indicates that a new context should be built because macros for travelling through the door have been discovered and these macros may facilitate the discovery of other macros. The new context will use the macros (M:1.01 and M:1.02) and generalize the actions by specified patterns. The knowledge indicates which patterns should be used to generalize some of the action steps. The braces around an action in the output trace show that a generalized action has been created. Dashes separate the parts of the pattern used. For example, (GRASP-DC*) indicates that the first part of the pattern matches GRASP exactly and anything after that is ignored by the don't care indicator (DC*). In the second context, the first macro discovered (M:2.01) is for moving to a box (at position x=0 or x=10), grasping the box, moving to and through the doorway (M:1.02), moving to a position to set the box down (x=11 or x=20), ungrasping the box, and returning through the door (M:1.01). Two macros with lower cognitive savings (M:2.03 and M:2.04) are also found. Cognitive savings (labelled cogsav in figure 3), is a numeric value representing the amount of savings one gains by using the substructure instead of the entities of the substructure themselves. Note that without the knowledge to generalize the actions, the system would not be able to discover the macro M:2.01 which defines most of the originally observed sequence of actions.

7 Future Work

Currently, the system does not recognize the significance of the x position in a move. Boxes that start at x=0 in the first room are always placed at x=11 in the second room. Also, the system does not understand that the last object grasped is the

---

*Figure 4 uses a PROLOG notation for describing the rules.
**The notation M:1.02 indicates the macro was found in the first context (1) and was the second macro discovered within the context (.02).
promotable-macro(X) <- macro-p(X), uses-door(X)

uses-door([X,Y]) <- move-through-doorway(X,Y)

uses-door([Xn]) <- uses-door(X)

move-through-doorway([move[Door-pos1], move[Door-pos2]]) <-
door-at(Door-pos1), door-at(Door-pos2),
not Door-pos1 = Door-pos2

deer-at([10,6])
deer-at([11,6])

Figure 4: Simple rules for macro promotion

same as the object most recently ungrasped. Although an EBL system given more precise rules about the semantics of the problem could determine this, a more effective and efficient manner of representing the macros, actions, and their related meanings needs to be developed.

Also, the system needs to discover plausible stopping criteria for the discovered loops and branching conditions for the conditionals. This involves maintaining information about the state of the world as actions are performed. By recognizing significant changes in the state at the end of a macro the system can propose that an interesting event of that state is the stopping condition. For example, if the system is stacking blocks to a specified height, as in the BAGGER system, then a block at the required height is the stopping condition. The stopping conditions need not always be direct components of the current state. A stopping condition could be a relation between a robot and an object in the world. The robot does not want to run into a wall, so it must move until it gets close to the wall. The position of the wall has not changed, but its relation to the robot has and if the robot moves again it will hit the wall. This type of stopping criteria is more difficult to discover.

In addition, a more complex method of proposing generalizations for the actions needs to be developed. One that uses constructive induction instead of rules coded for specific problems as done in current EBL systems is under consideration. This constructive induction could be done in an incremental manner, as done in LAIR [20] or a more knowledge-based approach could be accomplished by using the semantics of the actions and the effects of the actions on the current situation.

8 Conclusion

The PLAND system demonstrates how a learning system can incorporate background knowledge into many different levels in order to aid in the learning process. Each of these levels should use an integrated knowledge base to harvest the benefits offered from both EBL and SDBL systems. The multi-leveled approach to learning allows a system to use specific aspects of knowledge in unique ways at each level to guide the system to solutions more rapidly. There appears to be no a priori fixed number of levels that is optimal. Ideally a system could add and remove levels as needed. PLAND cannot do that, but it does allow the system to work on three different levels of generalization in the problem space and to work on different subproblems simultaneously.

9 Acknowledgements

This work benefited from discussions with Robert Stepp, Larry Holder, Bob Reinke, Bharat Rao, Diane Cook, and Bob Moll.

REFERENCES


J. W. Shavlik and G. F. DeJong, “An Explanation-Based Approach to Generalizing Number,” Proceedings of the Tenth International Joint Conference on Artificial Intelligence, Milan, Italy, August 1987, pp. 236-238. (Also appears as Technical Report UILU-ENG-87-2220, AI Research Group, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.)


