Leveraging Failures to Enhance Hierarchical Concept Learning when Training and Testing are Limited

Huzaifa Zafar & Daniel D. Corkill
University of Massachusetts Amherst
Computer Science Department
{hzafar,corkill}@cs.umass.edu

Abstract

Hierarchical concept learning constructs higher-level concepts using previously learned prerequisite concepts. We are working in an especially challenging context where only a small number of training instances for each concept are provided to the learning system. This limited instruction forces even the most skillful learner to make assumptions about the concept being taught—assumptions that can be incorrect. Given this uncertainty, multiple candidates may be proposed for the concept, each stemming from different assumptions that are consistent with the training.

We present a control strategy for managing the use of hypothesized concept candidates in higher-level learning. The strategy is based on three key ideas: 1) limiting prerequisite-candidate combinatorics by operating with a single selected candidate for each concept at any time, 2) using learning failure to select a different candidate for a direct or indirect prerequisite concept, and 3) using differences observed as candidates are used to guide candidate selection. We implemented and evaluated this novel Concept Candidate Management (CCM) strategy in MABLE, an electronic student that performs bootstrapped concept learning. Using the CCM strategy, MABLE learned concepts that were not successfully learned otherwise—without any additional training or testing and without any changes to learning algorithms.

Introduction

An automated system for hierarchical concept learning (Rivest and Sloan 1994; Nguyen et al. 2004) uses one or more learning algorithms ("learners") that learn higher-level *target* concepts using *prerequisite* concepts that were learned previously or provided to the system as primitive, base-knowledge concepts. We are working with an open, extensible learning system where any number of learners can be added to the system. When this system is given a new concept to learn, it is told which prerequisite concepts are to be used in learning the concept.[1] Concept instruction (in the form of training instances or other teaching modalities) is conveyed to the learning system using a formal *interaction language* (Mailler et al. 2009). When invoked to learn a target concept, a learner uses the instruction to incorporate (compose) the specified prerequisite concepts into a composable procedure performing the learned target concept.

Limited instruction & multiple concept candidates We are working in an especially challenging context where the learning system is provided with limited instruction (such as a few training instances) for each concept. This causes even the best of learners to make assumptions about the concept being taught (Hoffbeck and Landgrebe 1996; Zhou and Huang 2001; Zhang, Lin, and Zhang 2001). Those assumptions can be incorrect, forcing a learner to either hypothesize a single candidate for the target concept or avoid premature elimination of reasonable assumptions by proposing multiple candidates. Each of these proposed *concept candidates* stems from different assumptions made that are consistent with the instruction. As a very simple example, consider the blocks-world concept of a stack. A few labeled examples of stack and non-stack block arrangements are provided, all using blue blocks. A learner could assume that block color does not matter, that a stack must consist of blocks of the same color, or that stacks must be blue. Since a learner cannot eliminate any of these assumptions given the instruction, it could choose to propose a color-unimportant candidate, a monochrome candidate, and a blue-only candidate.

The main contribution of this work is a strategy for managing these multiple concept candidates effectively throughout open-ended concept learning (where new target concepts may be supplied and taught to the system at any time in the future). This strategy is important because multiple concept candidates are inherent given such limited instruction, no matter how much we try to improve the learners.[2]

The remainder of the paper is as follows. We first continue describing the challenging context imposed on our learning system and how failures can be recognized (even with very limited testing) and used to eliminate problematic concept candidates. Then we present our concept-candidate management (CCM) strategy and describe why it is effective.

[1] Therefore, the learning system does not need to identify the concepts to be learned or the precedence relationship between them, which is a major focus of work such as that of Datta and Kibler (Datta and Kibler 1993).

[2] For example, ensemble methods (Ali and Pazzani 1996; Dietterich 2000; Zhang et al. 2009) could be used to combine weak learners into a stronger learner, but given such limited training, multiple candidates representing alternative reasonable assumptions would still be needed.

Next we evaluate the performance of CCM in four learning domains as implemented in MABLE, an electronic student that performs bootstrapped concept learning. We conclude with a discussion of unexplored research directions and potential future work.

Leveraging Failures

In addition to the limited instruction provided for each concept, the learning system is given a *concept mastery exam* consisting of small number of test "questions" (as few as one).[3] A mastery exam involves answering a question or performing a skill that demonstrates the concept has been learned. For example a mastery exam question could be "Is this a stack?" Only a single candidate's answers to the exam questions for the target concept can be submitted by the learning system for grading. This one-time-only-per-concept grading is the only supervised testing that is available. Therefore, it is important that the learning system use grading to the best advantage.

The concept exam can be used to eliminate some of the candidates generated for the concept. If an answer is graded as incorrect, all candidates that produce that incorrect answer can be discarded. Answers graded as correct, on the other hand, only eliminate candidates if the question is known to have a single correct answer (based on the concept type or background knowledge), in which case all candidates that answered differently can be discarded.

Another constraint on our learning system is that its learners can use only the provided instruction for the concept. So, the graded answers and other candidate-use information (to be discussed later) cannot be given to a learner in an attempt to improve the candidates that it generates. The only change allowed is to invoke learners with the same instruction, but with different prerequisite concept candidates.

Prerequisite candidate use failures Candidates that are not eliminated by the concept exam can still cause problems when used to learn higher level concepts. Learners can fail to learn given a prerequisite candidate, producing no candidates for the target concept. Or, target candidates can be learned, but all of them are eliminated by the target concept's exam. Such prerequisite candidate *use failure* can be addressed only by learning the target concept again using a different candidate for one or more of the target concept's (direct or indirect) prerequisite concepts. An important insight in our work is that use failures provide an indirect avenue for eliminating problematic candidates. In other words, use failures (especially those encountered early in the use of a candidate) can be appreciated, rather than disparaged.

Compensatory learning Some learners are able to "fix" prerequisite-candidate problems by learning to compensate for the prerequisite's undesired behavior. For example, suppose a candidate for the concept of multiplying two integers incorrectly produces a negative result if both integers are negative. A learner, using the incorrect candidate when learning a higher-level arithmetic operation, could compensate by changing the sign of the result when both numbers are negative. Such *compensatory learning* can produce a short-term benefit of learning the target concept and even passing the concept exam. Unfortunately, compensatory learning masks use failures, allowing problematic candidates to be used longer than they would be otherwise. Because compensatory learning is internal to a learner, it cannot be prevented by any candidate-management strategy. Our CCM strategy, however, limits its effect by using candidates broadly, making it difficult for concept learners to compensate for problematic candidates consistently.

[3] Exam "questions" and "answers" to them are used loosely in this paper. A "question" might be a command to perform an action that demonstrates mastery of the concept and its "answer" would be the sequence of primitive low-level actions to be executed.

The CCM Strategy

The Concept Candidate Management (CCM) strategy controls the learning of a hierarchy of concepts in which multiple candidates are proposed by learners for each concept. The objective is to maintain a single selected candidate for each concept that does not have any observed failures in answering concept exam questions or when the candidate is used as a prerequisite in high-level-concept learning. Exploring all possible combinations of prerequisite concepts, and the concept candidates generated from them, is expensive. CCM limits this exploration without overlooking any viable candidates. The CCM strategy is based on three key ideas: 1) limiting the number of candidates generated by operating with a single selected candidate for each prerequisite concept at any time, 2) using failures in learning high-level concepts to select a different candidate for direct or indirect prerequisite concepts, and 3) using differences observed as candidates are used to guide candidate selection.

Differentiating candidates

CCM uses observed behavioral differences to guide the selection of candidates when answering concept exam questions and in selecting prerequisites for learning higher-level concepts. Every learned candidate takes the concept exam, and the answers are used to place every candidate into an equivalence set where the observed behavior of all candidates is identical.

Further differentiation can be made as candidates are used as a prerequisite of higher-level concepts. Although we would have liked to record all uses, in our setting we only had access to the inputs and outputs of prerequisite candidates calls when a high-level concept candidate answered an exam question. Each input given to the candidate was recorded as an unsupervised test question for the prerequisite concept, while the output was recorded as the candidate's answer. In addition, each newly recorded test question is answered by all other candidates of the prerequisite concept, and those answers are used to differentiate them further.

Leveraging learning failures

The CCM strategy leverages two kinds of use failures: exam failures and prerequisite use failures.

Exam failures When a concept is first learned, CCM selects a candidate at random for grading from the equivalence set with the most number of candidates. By selecting from the largest set, CCM maximizes the number of candidates that will be eliminated if an answer is graded as incorrect. If a submitted answer is graded as incorrect, the entire equivalence set containing the selected candidate is eliminated, and an alternate candidate is selected randomly from one of the remaining equivalence sets.

Prerequisite use failures When learning a target concept, the selected candidate for each of the target's prerequisite concepts is used by the learners. When a prerequisite use failure occurs, CCM goes into failure-resolution mode, changing one or more of the selected (direct or indirect) prerequisite candidates of the target concept being learned in an attempt to resolve the failure. Whenever a selected candidate is replaced with an alternate, all concept learning that used the previously selected prerequisite-concept candidate (directly and indirectly) will be discarded and those concepts must be learned again.

In failure-resolution mode, CCM first chooses the prerequisite concept in which to change the selected candidate. CCM prefers to choose the highest-level prerequisite concept that has previously unused candidates. Once all alternate candidate combinations have been tried at a level, alternate candidate combinations are tried for lower-level prerequisite concepts. This is for two reasons: 1) less work has been done with candidates at higher levels so fewer concepts have to be learned again and 2) the conjecture that the more a candidate has been used, the less likely it will cause a prerequisite use failure. If multiple prerequisite concepts are at the same level, ties are broken by preferring the prerequisite concept with the most number of equivalence sets containing no previously used candidates, since it provides more choices of alternate candidates to select from that are likely to behave differently when used. If ties still persist, one of them is chosen randomly.

When selecting an alternate candidate for a prerequisite concept, CCM selects one at random from a different equivalence set than any previously selected candidate. If all equivalence sets have at least one previously used candidate, the set with the most number of candidates that have not been determined to be unsuccessful (defined shortly) is selected first, and one of the previously unused candidates in that set is selected randomly. If all candidates have been previously used, one of those candidates is selected randomly. When a use failure occurs, CCM records the *combination* of prerequisite candidates as previously used and will not use that particular combination again. A prerequisite candidate can still be used, however, with different combinations of other prerequisite candidates.[4]

Once an alternate candidate is selected, higher-level concepts that were learned using the previously selected prerequisite candidate are learned again, potentially resulting in additional concept exam or prerequisite use failures. Those are resolved in the same way as the original use failure but within the initial failure-resolution mode. Once all use failures have been resolved, CCM exits failure-resolution mode and all candidates that do not use the current selected candidates as prerequisite are discarded.

If the use failure is resolved, all previously used combinations are determined to be *unsuccessful* and never used again. A concept candidate is unsuccessful based on use failures if all combinations of direct or indirect prerequisite candidates have been determined unsuccessful.

If CCM is unable to resolve a use failure, the concept that triggered the failure-resolution mode is determined to be a *failed concept* because no combination of prerequisites can be used to successfully learn the concept. The failed concept is removed from the concept hierarchy, and CCM continues learning as many other concepts as possible. In this case, CCM reverts to the same state of selected candidates from before entering failure-resolution mode (continuing as if the use failure never occurred).

Evaluation

Experiments with concept-candidate management strategies were conducted using MABLE (Modular Architecture for Bootstrap Learning Experiments) (Mailler et al. 2009), an electronic student for learning a curriculum of multi-level concepts.[5] Each learner in MABLE generated a single candidate for the target concept (a limitation we could not control).

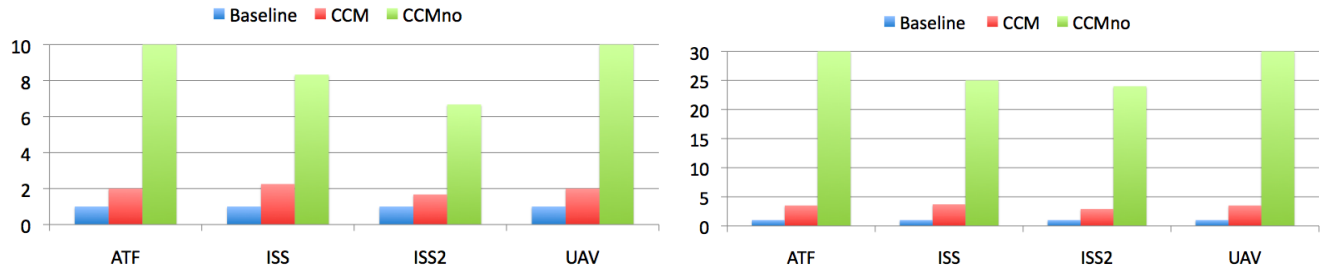
We compared the CCM strategy with three alternative strategies: 1) a baseline strategy that does not change the selected candidate once higher-level learning begins; 2) a "learner selects candidate" (LSC) strategy where each learner is allowed to select the candidate for each prerequisite concept that it uses when learning a concept; and 3) CCM using no candidate differentiation (CCMno) beyond the initial elimination of candidates based on the concept exam.

The baseline strategy The baseline strategy obtains answers to the mastery exam from each candidate generated for the concept by the learners. It then selects one candidate at random and submits its answers for grading, and the graded answers are recorded. If that candidate's answers pass the exam, it remains the selected candidate for the concept thereafter. If the candidate fails the exam, the selected candidate and all others whose answers were also incorrect are eliminated as unsuccessful. One of the remaining candidates, if any remain, is chosen at random to be the concept's selected candidate thereafter. If all candidates are eliminated as incorrect, learning of the concept fails.

The LSC strategy The LSC strategy is an improved version of the control strategy originally used in the MABLE electronic student. The original MABLE strategy eliminates only the concept candidate whose submitted answer fails the mastery exam. The improved LSC version eliminates all

[4] If a candidate is re-selected, previously learned high-level candidates that were learned using it and the other selected prerequisite candidates do not need to be learned again.

[5] We used curriculums, learners, and the control strategy from the 12/22/2010 version of MABLE for evaluation



(a) Number of candidate selections

(b) Number of times concept learning was performed

Figure 1: Additional Work Performed by CCM and CCMno Strategies (normalized to the baseline strategy)

candidates whose answers to the exam are incorrect (identical to the exam-failure elimination used in the baseline strategy). Both LSC and the original MABLE strategy allow each learner to select any single candidate (from those that have not been eliminated by the concept mastery exam) for each prerequisite concept when learning a higher-level concept. The graded mastery exam answers for each prerequisite concept are also available to the learner. The learner creates a single candidate for the higher-level concept (using any combination of prerequisite candidates it deems appropriate), but other learners can select different prerequisite candidates in producing their candidate for the same higher-level target concept.

Performance of each strategy

Table 1 shows the number of concepts learned successfully by MABLE using the baseline, LSC, and CCM strategies in each of four learning domains:

- ATF (Advanced Tactical Fighter), learning military formation strategies
- ISS (Instruction Set Simulator), learning to simulate computer machine instructions
- ISS2, learning to simulate a larger and more complex set of computer machine instructions
- UAV (Unmanned Aerial Vehicle), learning to operate a UAV for monitoring suspect behavior

The number of concepts learned unsuccessfully (i.e., all candidates learned for the concept were unsuccessful) and the number of concepts where learning failed (no candidates could be generated by the learners) are also shown. (Both CCM and CCMno learn the same concepts successfully, so only the CCM values are shown in the table.) The learners in MABLE did a fairly good job of generating successful candidates in all domains except ISS2, where the learners failed to generate any candidates (even unsuccessful ones) for over one-third of the concepts (this generation-failure count is shown in parentheses in the table). The additional runtime required by LSC and CCM compared to the baseline strategy is also shown. The additional runtime required with the LSC strategy is due to the extra work performed by each learner in selecting the prerequisite concepts used in learning. The additional runtime required with CCM stems from the work that must be performed when a candidate that has been used as a prerequisite becomes unsuccessful. This additional runtime with CCM includes the effort spent exhaustively searching all combinations of prerequisite candidates (direct and indirect) when no learner is able to generate a successful candidate using any of them.

Domain	Concepts Learned Successfully/Unsuccessfully (Failed)			Runtime Factor	
	Baseline	LSC	CCM	LSC	CCM
ATF	25/8 (5)	35/3	38/0	1.2	1.9
ISS	28/7 (2)	31/6	32/5	1.1	1.8
ISS2	10/13 (13)	13/11 (12)	14/10 (12)	1.2	3.0
UAV	40/7 (4)	48/3	50/1	1.3	2.0

Table 1: Concepts Learned in each Learning Domain

When the CCM strategy replaces a selected prerequisite candidate (direct or indirect), concepts that were learned using that unsuccessful candidate are learned again using each proposed replacement candidate. Figure 1 shows the total number of candidates that are selected by the baseline, CCM, and CCMno strategies and the number of times concept learning was performed. (The LCS strategy is not shown in Figure 1 because all candidate selection beyond that of the baseline strategy is performed internally by each learner and, as with the baseline strategy, concepts are not learned again.) In each learning domain, the CCM strategy performs nearly twice as many candidate selections as the baseline strategy (Figure 1(a)) and, due to learning again using newly selected prerequisite candidates, about four times as many concept learning invocations (Figure 1(b)). These totals exclude counts associated with the effort spent exhaustively searching all combinations of prerequisite candidates (direct and indirect) when no learner is able to generate a successful candidate using any of them.

Benefit of candidate behavior differentiation

Figure 1 shows that CCM is able to significantly reduce the amount of additional work, as compared with CCMno, by taking into account concept candidate behavior differences obtained as selected candidates are used as prerequisite candidates in higher-level concept tests. Figure 2 shows how candidate differentiation helps when selecting alternative candidates for a concept and when deciding which prerequisite concept should be considered first when the candidate of a higher-level concept becomes unsuccessful.

Selecting an alternative candidate for a concept from a different equivalence set than the one containing an unsuccessful candidate increases the likelihood that the differently behaving candidate will be successful. In each of the four

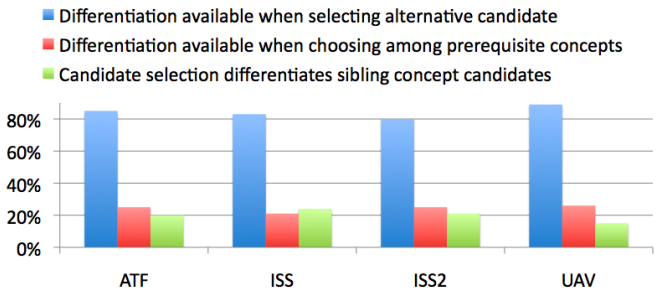


Figure 2: Effect of Differentiation in Selecting Alternate Successful Candidates

learning domains, observed behavior differences enable a successful alternative candidate to be selected from a different equivalence set about 80% of the time (as shown by the leftmost bar for each strategy). In the remaining 20% of the cases, no behavior differences had been observed to aid in selecting a successful alternative candidate (requiring a random search for successful alternatives from all remaining candidates).

As shown by the middle bar for each strategy, in approximately 20% of the cases when a choice needed to be made as to which prerequisite concept should be explored next for alternative prerequisite candidates, exactly one of the prerequisite concepts contained multiple equivalence sets. In these cases, that prerequisite concept was explored next and an alternate successful candidate was found in one of that concept's other equivalence sets.

Finally, as shown by the rightmost bar for each strategy, in approximately 20% of the cases when an alternate candidate was selected and used for learning, additional differentiating use cases were obtained for a sibling concept, further differentiating the sibling's prerequisite concepts.

Measuring extended use of training and tests

The CCM strategy takes advantage of higher-level concept tests and instruction to help identify unsuccessful (direct and indirect) prerequisite candidates and to differentiate them based on their use-case behavior. We next present measures of this extended candidate use.

Identifying unsuccessful candidates Figure 3 shows the amount of candidate use in higher-level training and tests that was required to identify them as unsuccessful. Figure 3(a) shows how many levels of higher-level use was required to identify a prerequisite candidate as unsuccessful. In all four domains, 70% of the unsuccessful candidates are identified by the concept exam and are not shown in the figure. The remaining 30% are shown in the figure, and the majority of those 30% were identified as unsuccessful when used directly as a prerequisite concept during learning or testing at the next level (+1L or +1T). However, the remaining candidates were identified as unsuccessful only when used indirectly as prerequisites at higher levels. The higher a prerequisite is used before it is identified as unsuccessful, the more work is required to select a successful alternative candidate and to learn concepts again using the newly selected prerequisite candidates.

Figure 3(b) shows how many uses of a prerequisite candidate (irrespective of usage level) was required to identify as unsuccessful the 30% that are not identified by the concept exam. Again, a majority of the remaining 30% were identified as unsuccessful within a few uses. However, close to 5% required 10 or more uses before they were identified as unsuccessful.

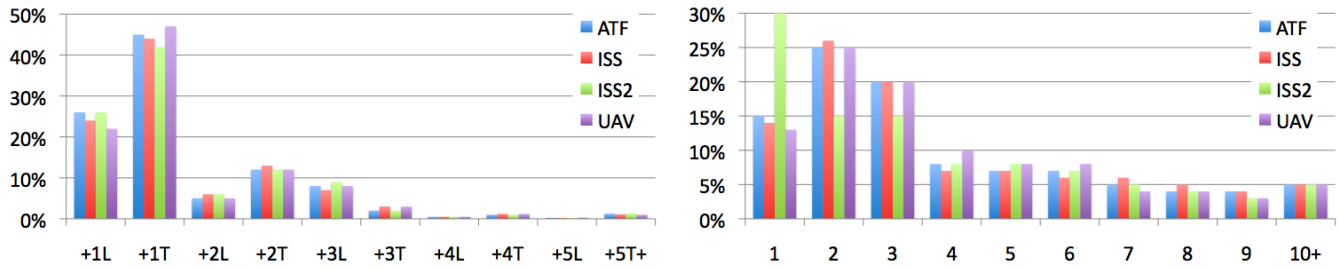
Identifying behavior differences Use of prerequisite candidates provides additional use-cases that can expose differences among all candidates of the prerequisite concept. Figure 4 shows how the use of a prerequisite candidate at higher levels and how the number of uses enabled further differentiation of candidates. In the ISS2 domain, the majority of the differentiation occurred with the concept exam. This is due to the large number of failed concepts where no candidates were generated.

Summary and Future Work

We presented Concept Candidate Management (CCM) as an effective strategy for managing the generation and selection of concept candidates using information gained as the candidates are used in learning higher-level concepts. By differentiating candidates based on their behavior and by associating unsuccessful learning of higher level concepts with the prerequisite candidates used to learn them, CCM can thoughtfully select alternative prerequisite candidates and control when unsuccessfully learned concepts have to be "learned again." Using the CCM strategy, MABLE learned concepts that were not successfully learned otherwise—without any additional training or testing and without any changes to learning algorithms.

The learners used in our MABLE-based CCM experiments were produced by other researchers and were used unmodified. (To us, these learners were truly "black boxes.") This limited exploration of several interesting research directions. First, uses of prerequisite candidates during learning was invisible to the CCM strategy (only the inputs and outputs as prerequisite candidates were used in answering higher-level concept examination questions could be recorded). We would have liked to obtain the inputs and outputs of prerequisite candidates as they were used by learners as they learned higher-level concepts. This additional source of candidate differentiation might improve CCM's ability to select successful alternative candidates. Second, each MABLE learner proposed only a *single* "best" candidate when invoked to learn a concept. If each MABLE learner could return multiple candidates for a concept—especially when the learner's confidence among those candidates is close—concepts that are currently unlearnable (due to the lack of a suitable prerequisite candidate) might be learned successfully given the additional candidates. Finally, if learners could annotate the candidates that they generate with their confidence in them, CCM could use those confidence values in candidate selection (replacing the random choices used in the absence of such information).

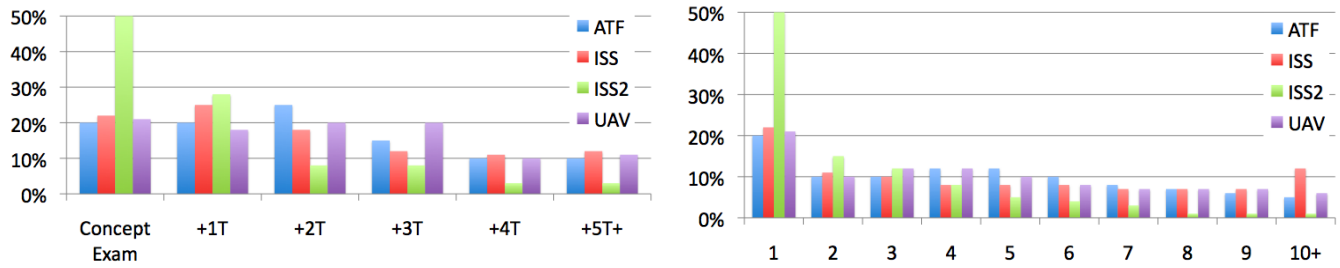
Another interesting, unexplored research direction is allowing a learner to use the inputs and outputs that CCM has recorded for a concept's candidates when the learner is in-



(a) Number of levels that a prerequisite is used

(b) Number of uses as a prerequisite

Figure 3: Percentage of Unsuccessful Prerequisite Candidate Detection by Use



(a) Number of levels that a prerequisite is used

(b) Number of uses as a prerequisite

Figure 4: Percentage of Prerequisite Candidate Differentiations by Use

voked to learn the concept again. This information could be used by the learner as an additional source of unsupervised instruction (Stone 2000; Whiteson and Stone 2003) stemming from how previously generated candidates were used in learning and testing higher-level concepts.

Acknowledgment

This effort was performed under the DARPA Bootstrapped Learning Program. It was sponsored in part by SRI International under agreement number 27-001251. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or SRI.

References

Ali, K. M., and Pazzani, M. J. 1996. Error reduction through learning multiple descriptions. *Mach. Learn.* 24(3):173–202.

Datta, P., and Kibler, D. 1993. Concept sharing: A means to improve multi-concept learning. In *In: Proc. of the 10th ICML*, 89–96.

Dietterich, T. G. 2000. Ensemble methods in machine learning. In *MCS '00: Proceedings of the First International Workshop on Multiple Classifier Systems*, 1–15. London, UK: Springer-Verlag.

Hoffbeck, J. P., and Landgrebe, D. A. 1996. Covariance matrix estimation and classification with limited training data. *IEEE Trans. Pattern Anal. Mach. Intell.* 18:763–767.

Mailler, R.; Bryce, D.; Shen, J.; and O'Reilly, C. 2009. MABLE: A framework for learning from natural instruction. In *AAMAS '09: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, 393–400. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

Nguyen, S. H.; Bazan, J.; Skowron, A.; and Nguyen, H. S. 2004. Layered learning for concept synthesis. In Peters, J. F.; Skowron, A.; Grzymala-Busse, J. W.; Kostek, B.; Swiniarski, R. W.; and Szczuka, M. S., eds., *Transactions on Rough Sets I*, volume 3100 of *Lecture Notes in Computer Science*, 187–208. Springer Berlin / Heidelberg.

Rivest, R. L., and Sloan, R. 1994. A formal model of hierarchical concept learning. *Inf. Comput.* 114:88–114.

Stone, P. 2000. *Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer*. MIT Press.

Whiteson, S., and Stone, P. 2003. Concurrent layered learning. In Rosenschein, J. S.; Sandholm, T.; Wooldridge, M.; and Yokoo, M., eds., *Second International Joint Conference on Autonomous Agents and Multiagent Systems*, 193–200. New York, NY: ACM Press.

Zhang, X.; Yoon, S.; DiBona, P.; Appling, D.; Ding, L.; Doppa, J.; Green, D.; Guo, J.; Kuter, U.; Levine, G.; MacTavish, R.; McFarlane, D.; Michaelis, J.; Mostafa, H.; Ontanon, S.; Parker, C.; Radhakrishnan, J.; Rebguns, A.; Shrestha, B.; Song, Z.; Trewhitt, E.; Zafar, H.; Zhang, C.; Corkill, D.; DeJong, G.; Dietterich, T.; Kambhampati, S.; Lesser, V.; and et al. 2009. An Ensemble Learning and Problem-Solving Architecture for Airspace Management. In *Proceedings of Twenty-First Annual Conference on Innovative Applications of Artificial Intelligence (IAAI-09)*, 203–210.

Zhang, L.; Lin, F.; and Zhang, B. 2001. Support vector machine learning for image retrieval. In *Proc. IEEE Int. Conf. on Image Processing*, 721–724.

Zhou, X. S., and Huang, T. S. 2001. Small sample learning during multimedia retrieval using biasmap. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 1:11.