Boosting-based Learning Agents for Experience Classification

Po-Chun Chen, Xiaocong Fan, Shizhuo Zhu, and John Yen
Laboratory for Intelligent Agents, College of Information Sciences and Technology
The Pennsylvania State University, University Park, PA 16802, USA
{pchen, zfan, szhu, jyen}@ist.psu.edu

Abstract

The capability of learning from experience is of critical importance in developing multi-agent systems supporting dynamic group decision making. In this paper, we introduce a hierarchical learning approach, aiming to support hierarchical group decision making where the decision makers at lower levels only have partial view of the whole picture. To further understand such a hierarchical learning concept, we implemented a learning component within the R-CAST agent architecture, with lower-level learners using the LogitBoost algorithm with decision stumps. The boosting-based learning agents were then used in our experiments to classify experience instances. The results indicate that hierarchical learning can largely improve decision accuracy when lower-level decision makers only have limited information accessibility.

1 Introduction

Experience acquisition and adaptation is tightly connected to learning. For example, instance-based learning [8] is particularly relevant to experience accumulation; the Recognition-Primed Decision (RPD) model [9] highly relies on the availability and correct recognition of past experiences. It is thus critical for agent architectures running in a dynamic environment to incorporate a learning framework and a flavor of "case-based reasoning."

The R-CAST agent architecture (RPD-enabled Collaborative Agents for Simulating Teamwork) [3] is built on top of the concept of shared mental models [1], the theory of proactive information delivery [4], and Klein's Recognition-Primed Decision (RPD) model [9]. The R-CAST agent architecture has implemented a "collaborative-RPD" decision process, which supports close human-agent collaborations in relevant information sharing, decision progress monitoring, and expectancy-based decision adaptation. R-CAST has been used in conducting experiments in simulated Command & Control domain and in studying research-related issues (e.g., [2]). However, R-CAST still lacks a built-in learning mechanism, assuming experiences are collected off-line from domain experts. This can limit the use of R-CAST when the collected experience is deficient or not directly applicable.

Since R-CAST is a team-based agent architecture, it is natural to develop learning approaches that can leverage team structures and teamwork knowledge associated with the R-CAST run-time environment. In this paper, we propose a learning approach that takes advantage of the hierarchical organization of agent teams for agent collaboration in decision making. Taking a group of distributed decision makers as an example, as a team, all the decision makers share the same goal and decision making context; however, they can propose different decisions due to their diversity in experience, knowledge, beliefs, perspectives, evaluation criteria, and other aspects. The idea of hierarchical learning is to organize an agent group such that a final decision maker can take advantage of the strength of diversity by combining decisions from lower-level agents with different criteria. Complementary to the shared mental model approach [12, 4], which states that proactive teammates can share information to establish global situation awareness, we focus on the fact that agents may unavoidably have to make decisions only based on partial information.

To realize the hierarchical learning concept, we extended the R-CAST agent with a learning component. By specifically focusing on classification problems, we chose boosting [10, 5, 6] as its core algorithm. Boosting has a number of variations, among which we chose LogitBoost [7] since it has been shown to be well-performed in multi-class classification problems. In addition, for flexibility and applicability, decision stumps (simple decision trees with only two leaves) was chosen as the base learners for the boosting algorithm since it has the least assumptions on underlying data models.

In the rest of the paper, section 2 presents the idea of hierarchical learning and the R-CAST learning component; section 3 describes three experiments; and section 4 concludes the paper.

2 The Learning Framework

In this section, we first introduce the concept of hierarchical learning, and then we briefly describe the learning component developed under the R-CAST architecture.

2.1 The Hierarchical Learning

The hierarchical learning is a variation of the original boosting. The idea is to build a multi-agent system to implement a committee of learners forming a multi-layered learning structure such as the example shown in Figure 1. An upper-level learning agent makes classification decisions based on the recommendations from its immediate lower-level agents. Learning agents at the same level can share the same goal and tasking context, but they typically have different perspectives and evaluation criteria. Consequently, they can generate different classification decisions regarding a same situation. Our assumption is that, by combining the diversity and learnabilities [10] of its lower-level agents, an upper-level agent could perform better than each individual of the lower-level agents.

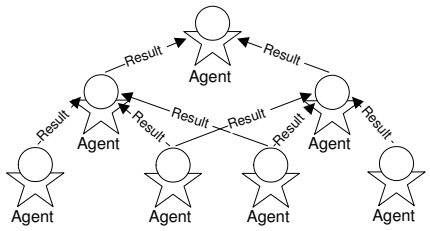

Figure 1. The hierarchical learning

This learning approach relies on the following layered training protocol:

(a) Each of the bottom-level agents is individually trained (can use independent training set) to obtain a classifier reflecting its decision-making criteria (perspective) regarding the domain problem.

(b) To train an upper-level agent, select a distinct set A of training instances, and request each of the immediate lower-level agents (already trained) to classify this set of instances. The classification results as well as the correct class labels are collected to form a set B of training instances for the upper-level agent. It is worth noting that training set B differs from set A. The instances in B take the lower-level agents' classification results as its attribute values. For example, if we have three agents named A1, A2, and A3, the composed training instances will be of the form *(result_of_A1, result_of_A2, result_of_A3, correct_class)*. The upper-level agent is then trained using set B, resulted with a classifier based on its children's results.

(c) Repeat the same procedure to train all agents level by level toward the topmost one. As a consequence, all the agents each will have a well-trained classifier that combines the perspectives of all its subordinates.

Obviously, the topmost agent can generate the final decisions considering all the contribution of the committee members. The performance can be sensitive to what strategies are used to balance the effects of each lower-level agent's contribution in producing the final decisions. For now, we adopt majority vote; in the future, we will explore other approaches such as weighted-sum and decision trees.

In general, this hierarchical learning approach provides a systematic and flexible way for organizing learning-enabled agents. Better performance can be gained by comprehensively considering the recommendations from a group of "weak learners."

This hierarchical learning is motivated by the original idea of boosting. Boosting [10] is a kind of ensemble learning which combines a number of weak learners to form a committee, and the committee integrates the decision by strategies such as majority vote [5] or weighted-sum computation. Although each of the weak learner can just be correct on certain fraction of the whole instance set, by boosting they can be coherently organized to provide quite desirable results.

However, the hierarchical learning differs from boosting algorithms (e.g., AdaBoost [6] and LogitBoost [7]) in the way how base learners are trained. Boosting builds the base learners by iteratively training a decision model, where each of the iteration creates a new learner of the same model, and the final decision is made by integrating these learners. Whereas in our approach, the base learners are individually trained agent entities; and the upper-level agents are trained level-by-level, exploiting the well-trained lower-level agents. In summary, this learning approach can be viewed as an agent-level boosting, where each learning agent may have its own distinctive knowledge, experiences, and evaluation criteria.

2.2 The Learning Component

In order to realize the hierarchical learning, we extended the R-CAST agent with a learning component. A knowledge base-centric framework was designed as shown in Figure 2, including five main functions working as follows. First, the preprocessing function prepares training instances by composing relevant pieces of facts of the type "TrainingInstance" to produce complete training instances. Second, the training function will build a classifier using the latest set of training instances. Third, whenever the agent recognizes a current situation, the classifying function will be invoked to perform classification and propose a recommendation with the latest trained classifier. Fourth, a scenario-

dependent verifier will be invoked to check the correct class label for the given situation. Last, the performance evaluator will determine the performance in terms of error rates.

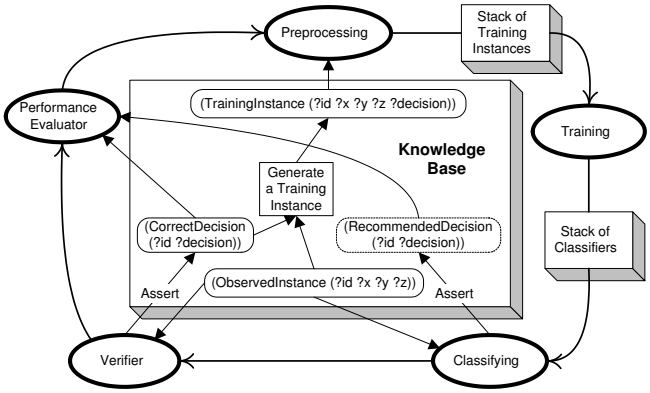

Figure 2. KB-centric learning cycle

In our implementation, LogitBoost [7] (with decision stumps) was adopted as the underlying classification algorithm. The Weka [11] data mining library was also used in our implementation, including the LogitBoost function and the instance handler.

3 Experiments

In order to evaluate the hierarchical learning, we conducted three experiments as follows: the first one has one single agent with complete training/testing instances (i.e., instances with all domain features available); the second one involves agents fed with partial instances where some features are missing; the third one consists of eleven agents, including ten "biased" learners from the second experiment and one upper-level agent which takes the biased learners' classification results as its inputting features.

To perform the experiments, we randomly generate instances using a simulated 5-class data model adapted from [7]. This model assumes all instances have ten attributes randomly drawn from a 10-D Gaussian distribution, and the decision boundaries are determined by properly choosing the thresholds $r = \sqrt{\sum_{i=1}^{10} x_i^2}$ such that approximately equal number of instances will be assigned to each class.

3.1 Experiment 1: Complete Training Instances

In experiment 1, the learner was always fed with complete instances. Using 5,000 independently drawn training instances and another 5,000 testing instances, the testing results were as shown in Figure 3. The error curve begins with 0.7532 and then gets lower and lower, stabilized after around 1,500 iterations. The best testing error rate is 0.1614, achieved by the classifier trained with 1,811 iterations. We attribute such a good performance to Boosting.

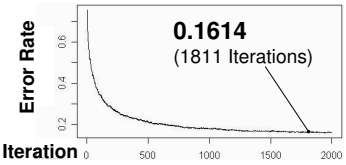

Figure 3. Experiment 1 — test error curve

3.2 Experiment 2: Partial Training Instances

In experiment 2, we set up ten learning agents, each handled instances with a different missing attribute. The same sets of complete training/testing instances were delivered to these ten agents, but they selectively ignored the values of their respective missing attribute. By training each agent's classifier with 5,000 independently drawn instances and testing with another 5,000 ones, the results were as shown in Figure 4. The error rates rose more than twice as compared with experiment 1. We can attribute such performance decrease to the use of partial instances.

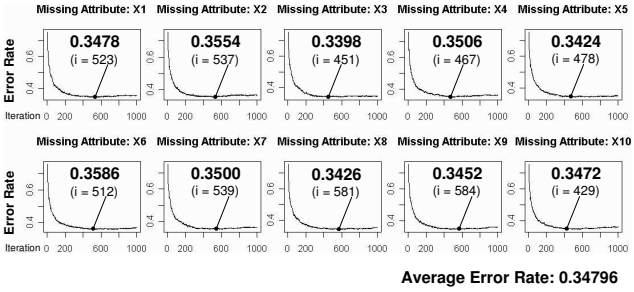

Figure 4. Experiment 2 — test error curves

3.3 Experiment 3: The Hierarchical Learning

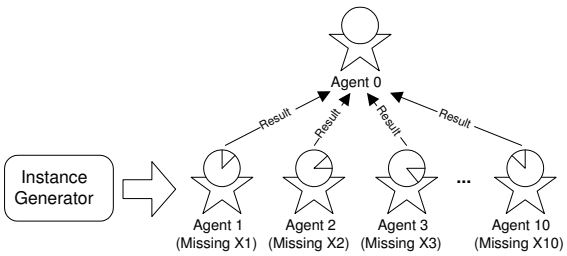

Figure 5. Experiment 3: hierarchical learning

In this experiment, as shown in Figure 5, we used a 2-layered structure to organize 11 agents, where *Agents 1..10*

were trained with partial instances, and *Agent 0* treated the classification results of *Agents 1..10* as input attributes to produce the final classifications. Instances used by *Agent 0* are in the format of (ID, result of *Agent 1*, .., result of *Agent 10*), in which the attributes other than ID are generated and delivered by the bottom-level agents in real-time. For the upper-level agent to make a final decision, we adopted majority vote as its decision strategy.

We performed classification on the same set of 5,000 testing instances used in the previous experiments, resulting an error rate of 0.2612, which is only three fourth of that produced by a lower-level agent.

3.4 Result Analysis

The results of each experiment using the same testing instances are summarized in Figure 6 (the solid lines). The error rate can be lowered to 0.2612 if an additional agent is employed to coordinate the "biased" agents. It is worse than one agent learning with complete instances by 61.8% but better than those with partial instances by 24.9%.

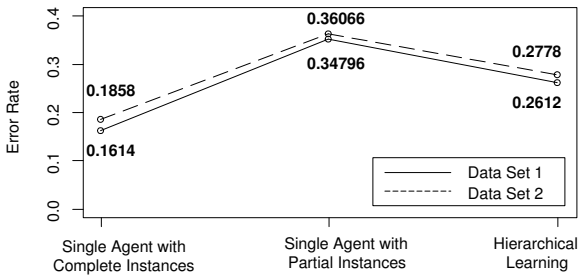

Figure 6. Results of the experiments

The dashed lines in Figure 6 show the result of another series of experiments which use the same set of agents to do classification on a new simulated data set with 5,000 testing instances. The result is consistent with the previous experiments. The performance of hierarchical learning is worse than one learning agent using complete instances by 49.5% but better than those using partial instances by 23.0%. Learning with missing attributes would lead to higher error rates, but the error rate can be reduced by applying a second level of coordination on these agents.

The hierarchical learning is based on other agents' classifications to produce the final result. A higher-level agent can sometimes be misled by the others if most of the lower-level agents mis-classify the instances. Although the hierarchical learning does not perfectly work all the time due to either other agents' errors or its own mistakes, the overall performance is still good enough. The error rate is expectably worse than agents using complete instances but significantly better than those using partial instances.

4 Conclusion

In this paper, we described the approach of hierarchical learning for supporting collaborative decision making. Then we conducted three experiments using learning-enabled R-CAST agents to evaluate the performance of this approach. Our results show that it can significantly improve the outcome of a group of biased learners.

The hierarchical learning concept can also be viewed as a way taking advantage of the strength of diversity. The results encourage us to further discover the potential theoretical roots. For future work, we also plan to apply this concept to enhance group-decision making and experience acquisition within a multi-context dynamic environment.

References

[1] J. A. Cannon-Bowers, E. Salas, and S. Converse. Cognitive psychology and team training: Training shared mental models and complex systems. *Human Factors Society Bulletin*, 33:1–4, 1990.

[2] X. Fan, B. Sun, S. Sun, M. McNeese, and J. Yen. RPD-enabled agents teaming with humans for multi-context decision making. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 34–41, Hakodate, Hokkaido, Japan, May 2006.

[3] X. Fan, S. Sun, M. McNeese, and J. Yen. Extending recognition-primed decision model for human-agent collaboration. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 945–952, The Netherlands, July 2005.

[4] X. Fan, J. Yen, and R. A. Volz. A theoretical framework on proactive information exchange in agent teamwork. *Artificial Intelligence Journal*, 169(1):23–97, 2005.

[5] Y. Freund. Boosting a weak learning algorithm by majority. In *Information and Computation*, volume 121, pages 256–285, 1995.

[6] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[7] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 28(2):337–407, 2000.

[8] C. Gonzalez, J. F. Lerch, and C. Lebiere. Instance-based learning in dynamic decision making. *Cognitive Science*, 27(4):591–635, 2003.

[9] G. A. Klein. The recognition-primed decision model. In *Sources of Power: How People Make Decisions*, chapter 3, pages 15–30. The MIT Press, 1998.

[10] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[11] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, 2 edition, 2005.

[12] J. Yen, X. Fan, S. Sun, T. Hanratty, and J. Dumer. Agents with shared mental models for enhancing team decision-makings. *Journal of Decision Support Systems*, 41(3):634–653, 2006.