Sharing a concept

Sandip Sen and Parijat Prosun Kar
Mathematical & Computer Sciences Department
University of Tulsa
e-mail: sandip@kolkata.mcs.utulsa.edu, karpa@euler.mcs.utulsa.edu

Abstract

Though various interesting research problems have been studied in the context of learning agents, few researchers have addressed the problems of one, knowledgeable, agent teaching another agent. Agents can do more than share training data, problem traces, learned policies. In particular, we investigate how an agent can use its learned knowledge to train another agent with a possibly different internal knowledge representation. We have developed an algorithm that can be used by a concept learning agent, the trainer, to iteratively select training examples for another agent, the trainee, without any assumptions about its internal concept representation. We present initial results where the trainer agent is an instance based concept learner and the trainee agent is a decision tree learner.

Introduction

Transfer of learning from one human to another is a laborious and fragile process. The success of such a transfer of knowledge or expertise depends on a variety of factors including common background knowledge of teacher and learner, the teaching skill of the trainer, the learning skill of the learner, the availability of original training information and context, etc. On the other hand, it is often argued that the transfer of learning between software agents is instantaneous and fail-safe and, therefore, "if one computer learns something, all of them have learned it!"

Though there is some merit to this argument in an ideal world, if we consider the real world, one can find convincing evidence that such a blanket statement is simply not justifiable. In particular, any time two computer systems or software agents use different internal knowledge representations or algorithms, the transfer of knowledge from one to another cannot be accomplished by a simple copy operation. Consider, for example, one software agent using a rule based system for learning and reasoning whereas another using a neural network representation. The process of transferring learned knowledge from one to the other can immediately be recognized as a challenging procedure. Such a knowledge transfer can well turn out to be as complex and error-prone as is the transfer of knowledge between two human beings.

In this paper, we address the problem of transfer of knowledge between a trainer and a trainee agent. The knowledge being transferred is a concept description. A concept description is a boolean-valued function that classifies input examples as members or non-members of the target concept (Mitchell 1997). We assume that the trainer agent does not have access to the internal knowledge representation of the trainee agent, but can observe its concept recognition abilities by providing exemplars and non-exemplars of a concept and observing the performance of the trainee agent. Though there can be additional research issues that augment the transfer of concept description knowledge between these two agents, we focus on the incremental selection of training examples by the trainer to expedite the learning of the trainee agent.

The primary advantage of concentrating on the iterative selection of training examples from observed performance is that the basic process can be independent of the internal knowledge representations and algorithms of the trainer and learner. A secondary benefit is that the developed procedure can:

- be motivated by procedures used by human teachers to choose training exemplars for human students;
- be used, in principle, by a software agent to teach a human user.

Though these aspects are not our primary concern, it does open up fruitful avenues for future work.

While a number of researchers have looked into the research issues involved in agents learning concurrently or cooperatively, there is little work on a trainer agent carefully choosing training instances based on past performance of the trainee agent. In this paper, we first present the general architecture of the Agent Teaching Agent (ATA) framework. Though it is not necessary, we do cover the learning process of the teacher, i.e., how the teacher itself acquired the knowledge it is now trying to teach the trainee. When the teacher agent assumes the teaching role, we assume that the entire set of training instances that it experienced is no longer available to teach the trainer. While this assumption can be violated in particular real-life situations, it allows us to develop more general-purpose training framework that does not depend on the availability of original training instances. This assumption is also justified if:

- the original training set size was too large to be carried around by a compact-sized learner, e.g., mobile agents, or
- the teacher had no reason to believe that it will need those training instances in the future and hence got rid of them after its own learning was over, e.g., the teacher did not foresee its teaching role.

We have run some initial experiments with the teacher using an instance-based learning algorithm, IB2 (Aha, Kibler, & Albert 1991), and the learner using a decision-tree based learning algorithm, C4.5 (Quinlan 1993). For initial development and evaluation we decided to use a set of artificial concept descriptions that we have used in our previous work (Sen & Knight 1995). We have also started evaluating our methodology on some data sets from the UCI machine learning repository (Murphy & Aha 1992). Though these latter data are not included in the current paper, we plan to include them in the next version.

In the rest of the paper we first present the ATA framework with details of the algorithm used for incremental training set determination. We then illustrate the target concepts used for evaluation. This is followed by the experimental results and some observations on our experiments. Finally, we discuss relevant existing literature, present the summary conclusions from our initial experiments and identify currently ongoing and planned work.

Agent Teaching Agent Framework

The motivation in our work has been that a teacher agent can guide the learning process of a learner agent by observing the latter's problem solving performance. In the context of concept learning, this means that based on the success and failure of the trainee in classifying given exemplars, the trainer can choose an appropriate sequence of training examples to guide the learning process of the trainee.

The basic ATA framework architecture is presented in Figure 1. The Trainer agent first acquires the target concept from its interaction with an environment, and using its learning module. This learning process produces a target concept description in the internal knowledge representation format of the trainer agent. These formats can range from logical rules (Quinlan 1990), genetic structures (DeJong 1990), computer programs (Koza 1992), decision trees (Quinlan 1986), stored instances (Aha, Kibler, & Albert 1991), neural networks (Rumelhart, Hinton, & Williams 1986), etc. The trainer agent also has a training module which interacts with the trainee module and provides successive training and testing set to train and evaluate the progress in learning of the trainee agent. The trainee learns its own concept description from the set of classified training examples provided by the trainer. It also classifies each of the unclassified test examples provided by the trainer and returns these classified instances to the trainer for evaluation.

We envisage an iterative training procedure in which alternatively the trainer selects a set of training and testing exemplars, the trainee trains using the training set and then classifies the testing set, the trainer observes errors made by the trainee in classifying the instances in the last testing set and accordingly generates the next training and testing set. This iterative process converges when trainee error falls below a given threshold. We present these iterative training steps in an algorithmic form in Figure 2.

The algorithm presented in Figure 2 needs to be further fleshed out to realize an actual implementation. In particular, we have to specify procedures for selection of the initial training and testing sets, N_0 and T_0, and the generation of the next test set T_{i+1} based on the mistakes, M_i, made by the trainee on the current test set.

We first present the underlying principles for designing these procedures. When selecting the initial training and testing instances, the goal is to select the most discriminating examples that help identify regions of the input space that do and do not belong to the target concept. For example, if a hyperplane separates instances of the target concept from non-instances, then points close to and on both sides of that hyperplane may be selected as initial training and testing set members. When selecting the next set of training and testing instances, the goal is to first isolate the mistakes made on the previous test set, and for each of these instances, find a few neighboring points, using some of them as training data and the rest as test data. Note that the true classification of these points are not known in general, and only their estimated classification, based on the concept description knowledge previously acquired by the trainer, can be used.

The actual procedure for selecting the sets N_0 and T_0 will depend on the internal representation used by the trainer agent. In our experiments, described in the following section, we have used an instance based learner as the trainer. This learner stores a subset of the training examples as its concept description. To select N_0 and T_0, we first sort, in increasing order, the stored points by the nearest distance to another stored point of the opposite classification. So, the sorting metric for any stored instance s in the set of stored instances, S, is

$$\min_{r \in S \wedge c(s) \neq c(r)} d(r, s),$$

where $d(r, s)$ is the distance between the points r and s in the input space, and $c : X \rightarrow \{0, 1\}$ is the boolean concept description function that classifies any instance in the input space X as a member or non-member of the target concept. Let this sorted list, L, be further divided into L_1 and L_0, viz., lists containing members and non-members of the target class.

We decided to start with a percentage of stored points

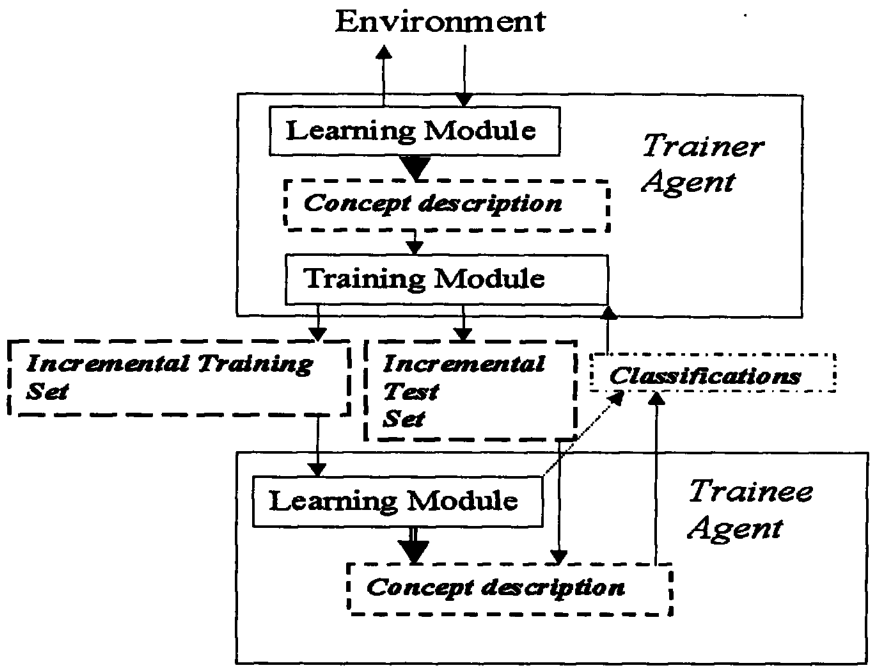

Figure 1: Agent Teaching Agent (ATA) framework.

as training and test set. In our experiments we started with 40% of stored points as the initial training set and 20% of the stored points as the initial test set. Equal number of instances of both classes were chosen for the initial test and training sets by going down the sorted lists L_0 and L_1.

The actual procedure for selecting T_{i+1}, the next set of test examples, from M_i, the mistakes made in the previous cycle, was more general. For each point $m \in M_i$ we randomly generated n points within a square of sides ϵ centered at m. Each of these $n * |M_i|$ points were classified by the trainer's learned concept description function $c_{trainer} : X \rightarrow \{0, 1\}$. This portion of the instance selection process is generic and can be reused by trainers which are not instance-based in nature. For each $m \in M_i$, we also locate the nearest stored point of the same category which have not already been used as a test or training instance in a previous iteration. The motivation was to see if the trainee continues to make mistakes on similar points. This portion of the instance selection process is specific to learners who store some of the training instances. A union of these two set of instances is then used to test the trainee in the next training iteration.

Experimental setup

As mentioned above, in the preliminary set of experiments we have run, the trainer agent is an instance-based learner. We have used the IB2 algorithm as the learning module of the trainer agent. IB2 is an incremental algorithm that receives one instance at a time and stores only those instances which it cannot correctly classify using prior stored instances (Aha, Kibler, & Albert 1991). The performance of this algorithm and the number of points it stores depends on the order of arrival of the input instances, e.g., if all instances of the same class precedes all instances of the other class the performance will be poor as only the first instance of the initial class will be stored.

For the trainee agent, we used the C4.5 algorithm, a batch-mode decision tree learner (Quinlan 1993). This algorithm is one of the most often-cited work in concept learning and classification literature in machine learning and has robust performance in the presence of noise.

For evaluation of our ATA framework we decided to use a suit of concept description problems for which we can easily vary the concept complexity (Sen & Knight 1995) (see Figure 3). For each problem, there are two continuous-valued input variable, and the input space is the unit square, i.e., each variable ranges between 0 and 1. The non-shaded regions belong to the target concept. For the different problem instances, the input is fragmented into different number of regions and this varies the complexity of the learning problem. The problems are somewhat difficult for the decision tree learning algorithm as decision surfaces are not axis-parallel. In addition, we also experimented with a *circle* domain where any point outside a circular region of radius 0.25 and centered at (0.5,0.5) belongs to the target concept. This was chosen to experiment with non-linear decision surfaces. We are also experimenting with a real-life domain with data from breast-cancer patients that we obtained from the UCI machine learning repository.

At the present time, however, we have only completed

```
Procedure Train-Agent(Trainer, Trainee, Trainer-knowledge)
{
        Select initial training set, $N_0$, and initial testing set, $T_0$, from Trainer-knowledge;
        $i \leftarrow 0$;
        Repeat{
                Train Trainee on training set, $N_i$;
                Let $M_i$ be the instances in $T_i$ misclassified
                by Trainer after training on $N_i$;
                Generate next test set, $T_{i+1}$, based on $M_i$ and Trainer-Knowledge;
                $N_{i+1} \leftarrow N_i \cup T_i$;
                $i \leftarrow i + 1$;
        } until($|M_i|$ < threshold);
}
```

Figure 2: Algorithm for generating training and testing sets.

experiments with the 2/1 domain. For each domain we generated 1000 points from a uniform distribution over the input space. We then ran experiments with five-fold cross-validation, i.e., the results were averaged over five training-test set pairs. The size of each training set was 800 and that of each test set was 200.

In the experiments, for each mistake made by the trainer, 3 points are generated in the square of side 0.1 with the error point at the center, i.e., $n = 3$, $\epsilon = 0.02$. We stopped the training loop when the trainee made no mistakes on the last incremental test set.

The performance of the trainee over the incremental test set is not representative of the overall quality of the learned knowledge. To evaluate the progress of learning of the trainee agent over successive iterations of training we use the learned concept description at the end of each iteration to classify the entire test of 200 instances. The corresponding performance over the entire test set gives us a reasonable estimate of the generality of the learned knowledge.

Initial Results

We plot the classification accuracy of the trainee agent over the entire test set, with the average and standard deviation over the five cross-validation runs, for the 2/1 problem in Figure 4. For comparison we also include a line that denotes the average performance of C4.5 if it had access to the entire training set of 800 training instances. We see that the ATA framework does lead to a quick improvement in performance of the trainer and reaches approximately 88% accuracy with a relatively small number of training instances. We ran another set of experiments where randomly selected points were classified by the trainer using its learned knowledge and then presented as testing instances in the iterative training phase. The increase in performance of the trainee with increase in training instances was significantly worse compared to the ATA framework which chooses instances based on the mistakes made in the previous training iteration.

We also evaluated the trainee's performance by presenting it with all the instances that were stored by the IB2 learning algorithm, i.e., the entire learned knowledge of the trainer, in one shot. The IB2 algorithm stored on the average about 50-60 of the 800 training instances presented to it. The performance of the trainee agent using the C4.5 algorithm, when trained on these few instances, resulted in only about 75% accuracy on the entire test set.

We wanted to test our approach on larger, real-life data sets. As a start, we have run experiments on the *Spamabse database* obtained from the Machine Learning Repository hosted by the CS department of the University of California Irvine (verb+http://www1.ics.uci.edu/ mlearn/MLSummary.html+). This data set has 4601 instances and 57 continuous valued attributes. The goal of the concept learning is to identify a given e-mail as spam or not based on features that counted the percentage of times certain words or characters occurred in the e-mail. Spam can originate from diverse sources like advertisements for products or web sites, make money fast schemes, chain letters, pornography etc. The collection of spam e-mails came from our postmaster and individuals who had filed spam. The collection of non-spam e-mails came from filed work and personal e-mail. The mails are classified into two classes, spam and non-spam e-mail. We performed 5-fold cross validation on this data set. IB2 typically stored between 300 and 600 training instances. The average test accuracy of IB2 and C4.5 algorithms on this data, when trained on 80% of the instances and averaged over the 5-fold cross-validation, are 83.2% and 86% respectively. Now with the incremental training scheme in the ATA framework, we got an average testing accuracy of about 80% with C4.5. This result was obtained with the C4.5 algorithm being provided with significantly lower number of training data than the entire training set. In this domain, the test performance of C4.5 was somewhat higher when it was trained on the entire set of points stored by IB2. This suggests that the incremental training and test set selection algorithm in ATA can be further improved. We should

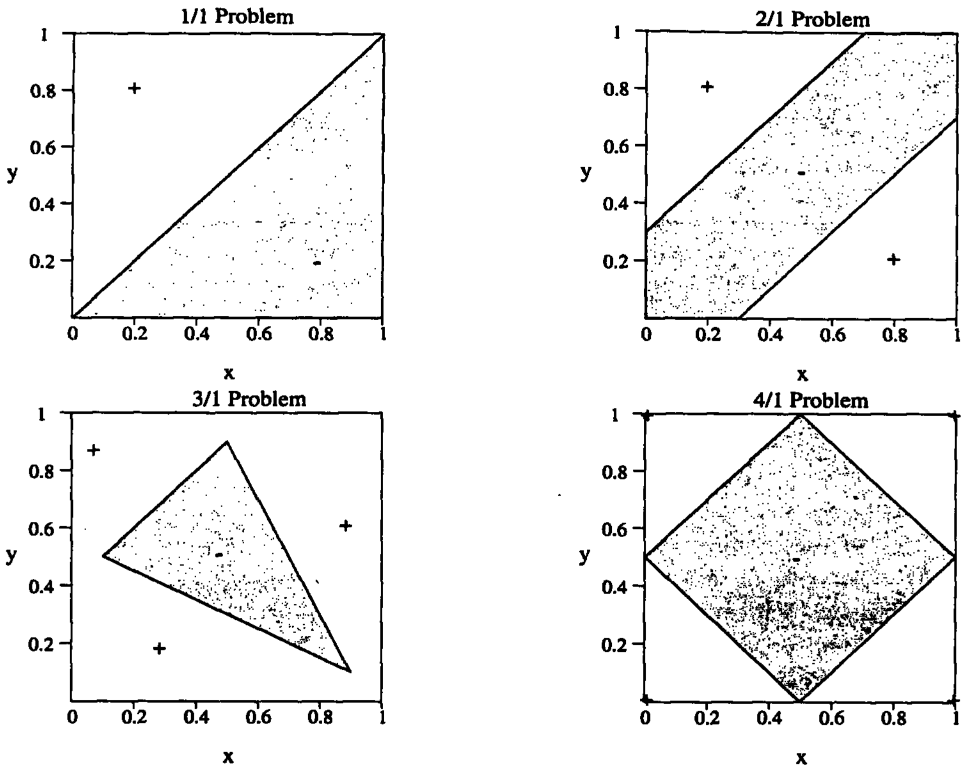

Figure 3: The classification problem set.

note, however, that training points will not be stored in general with other learning algorithms, e.g., if C4.5 is used as the trainer agent.

Related Work

Multiagent learning has been an active area of research in the late 90s (Sen & Weiß 1999; Stone & Veloso 2000). Though there have been considerable research on agents learning concurrently or cooperatively (Prasad, Lesser, & Lander 1996; Provost & Hennessy 1996), or even learning by observing opponent's behavior (Carmel & Markovitch 1996; Hu & Wellman 1998; Littman 1994), there has been little work in which an agent proactively trains another agent. The most relevant work comes from one learner telling another agents what portions of the search space to ignore (Provost & Hennessy 1996), a learner sharing experience (Banerjee, Debnath, & Sen 2000), problem-solving traces or even learned policies (Tan 1993) with another concurrent learner. Tan's work (Tan 1993) of an expert sharing effective problem solving traces with a novice agent and Clouse's work of a trainer suggesting actions to take (Clouse 1995) are perhaps the closest in motivation to the current work, but the iterative nature of teaching, at the heart of the ATA framework, is not addressed by them.

The other issue we need to address is the generality of the current framework. As noted before, the selection of the initial training and test set was facilitated in our experiments by the choice of an instance based approach as the learning module in the trainer. We need to work on using other knowledge representation schemes to extract decision surfaces learned from the training data. While some recently developed algorithms like Support Vector Machines (Burges 1998) lend itself easily to such introspection, more involved processing would be needed to work with opaque learned knowledge like neural networks (though some work has been done to interpret neural networks (Towell & Shavlik 1992)). As a brute force method, however, we can always generate a number of points, classify them with the trainer's learned concept description and use those points which were close to points of the opposite class as the initial training and test sets provided to the trainee agent. Such point generation need not be completely arbitrary, and can be focused on regions which shows mixture of exemplars and non-exemplars. Other, more knowledge-directed approaches can be studied in the context of symbolic classification knowledge produced by decision tree and rule learners.

Conclusions

We have proposed an ATA framework in which a trainer agent can used its learned concept description to iteratively generate training examples to train a trainee agent. No assumption is made about the internal knowledge representation and learning procedure of the

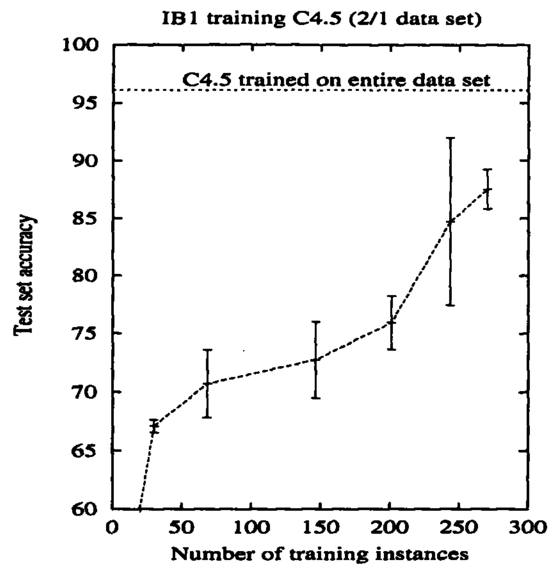

Figure 4: Trainer performance on test set in the 2/1 problem with increasing training iterations.

trainee agent. We present arguments for the generality of our approach and evaluate it in a sample artificial domain with an instance-based learner as the learning module of the trainer and a decision tree learner as the learning module of the trainee agent. Initial results are encouraging and demonstrates effective learning of the trainee agent.

We are currently running experiments on a wider set of problem instances that contain both artificial and real-life data and plan to include the results in the next update of this paper. We are also planning to use a support-vector machine based trainee agent, which will be trained by both the current instance-based trainer agent and a decision-tree based trainer agent. We have developed an initial sketch of how a decision-tree based trainer agent can generate effective sequences of training data and are looking forward to evaluating that approach.

Though the results presented here are preliminary in nature, we believe the ATA framework holds promise, and the trend shown in the initial results will generalize and scale well.

References

Aha, D. W.; Kibler, D.; and Albert, M. K. 1991. Instance-based learning algorithms. *Machine Learning* 6(1).

Banerjee, B.; Debnath, S.; and Sen, S. 2000. Combining multiple perspectives. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 33–40. San Francisco: CA: Morgan Kaufmann.

Burges, C. J. C. 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery* 2(2):121–167.

Carmel, D., and Markovitch, S. 1996. Learning models of intelligent agents. In *Thirteenth National Conference on Artificial Intelligence*, 62–67. Menlo Park, CA: AAAI Press/MIT Press.

Clouse, J. A. 1995. Learning from an automated training agent. ML-95 Workshop on Agents that Learn From Other Agents.

DeJong, K. A. 1990. Genetic-algorithm-based learning. In Kodratoff, Y., and Michalski, R., eds., *Machine Learning, Volume III*. Los Alamos, CA: Morgan Kaufmann.

Hu, J., and Wellman, M. P. 1998. Multiagent reinforcement learning: Theoretical framework and an algorithm. In Shavlik, J., ed., *Proceedings of the Fifteenth International Conference on Machine Learning (ML'98)*, 242–250. San Francisco, CA: Morgan Kaufmann.

Koza, J. R. 1992. *Genetic Programming: On the programming of computers by means of natural selection.* Cambridge, MA: MIT Press.

Littman, M. L. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, 157–163. San Mateo, CA: Morgan Kaufmann.

Mitchell, T. 1997. *Machine Learning*. Boston, MA: WCB McGraw-Hill.

Murphy, P., and Aha, D. 1992. UCI repository of machine learning databases [machine-readable data repository].

Prasad, M. N.; Lesser, V. R.; and Lander, S. E. 1996. Learning organizational roles in a heterogeneous multi-agent system. In *Proceedings of the Second International Conference on Multiagent Systems*, 291–298. Menlo Park, CA: AAAI Press.

Provost, F. J., and Hennessy, D. N. 1996. Scaling up: Distributed machine learning with cooperation. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 74–79. Menlo Park, CA: AAAI Press.

Quinlan, R. J. 1986. Induction of decision trees. *Machine Learning* 1:81–106.

Quinlan, R. J. 1990. Learning logical definitions from relations. *Machine Learning* 5(3):239–266.

Quinlan, R. J. 1993. *C4.5: Programs for Machine Learning*. San Mateo, California: Morgan Kaufmann.

Rumelhart, D.; Hinton, G.; and Williams, R. 1986. Learning internal representations by error propagation. In Rumelhart, D., and McClelland, J., eds., *Parallel Distributed Processing*, volume 1. Cambridge, MA: MIT Press.

Sen, S., and Knight, L. 1995. A genetic prototype learner. In *Proceedings of the International Joint Conference on Artificial Intelligence.*

Sen, S., and Weiß, G. 1999. Learning in multiagent systems. In Weiß, G., ed., *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.* MIT Press. chapter 6, 259–298.

Stone, P., and Veloso, M. 2000. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots* 8(3).

Tan, M. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning*, 330–337.

Towell, G., and Shavlik, J. 1992. *Interpretation of artificial neural networks: Mapping knowledge-based neural networks into rules.*