Proceedings of the 2000 IEEE/RSJ
International Conference on
Intelligent Robots and Systems

Expertness Measuring in Cooperative Learning

Majid Nili Ahmadabadi * Masoud Asadpur * Seyyed H. Khodaabakhsh* Eiji Nakano**

*Robotics Laboratory, Faculty of Engineering, University of Tehran
*School of Intelligent Systems, Institute for studies on Theoretical Physics and Mathematics
**Advanced Robotics Lab., GSIS, Tohoku University, Japan
E-mail: nily@sofe.ece.ut.ac.ir, majid,nakano@robotics.is.tohoku.ac.jp

Abstract

Cooperative Learning in a multi-agent system can improve the learning quality and learning speed. The improvement can be gained if each agent detects the expert agents and use their knowledge properly. In this paper, a new cooperative learning method, called ***Weighted Strategy Sharing (WSS)*** *is introduced. Also some criteria are introduced to measure the expertness of agents. In* ***WSS****, based on the amount of its teammate expertness, each agent assigns a weight to their knowledge. These weights are used in sharing knowledge among agents in our system.* ***WSS*** *and the expertness criteria are tested on two simulated Hunter-Prey problem and Object Pushing systems.*

1 Introduction

Learning is an essential part of Intelligent Behavior; if a creature has no learning capability, it can not adapt itself to the environment changes. All creatures do not need to learn all things individually and search to find solution of problems independently. Agents can communicate knowledge and information with each other and take advices or training commands from other agents, experts or even non-expertness. This is a kind of cooperation we call it cooperative *learning* and define it as follows:

Any method used by the learner agents through cooperation with their teammates to improve the learning velocity or quality, is a way of Cooperative Learning.

It is noteworthy that, cooperative learning needs communication. But communication could be implicit and nonverbal.

Until now, cooperative learning has not been investigated deeply. In almost all of multi-agent learning papers, cooperation of agents in learning is not complete or cooperation is uni-directional between a trainer agent and a learner. Because of having more knowledge acquisition resources, cooperation in learning in multi-agent systems may result in a higher efficiency compared to individual learning. Researchers have shown improvement in learning when agents cooperate to learn [Tan,1993].

Cooperative Learning is observed in human and some animal societies. News broadcasting, information distributing, publishing books, magazines and newspapers, imitation, consulting, training, rewarding, punishing, advising and so on, are some methods of cooperative learning in human groups.

Also, ants and bees are excellent samples of showing cooperative learning in finding shortest path to food spots or home [Dorigo&Gambardella,1997]. Also in Immune System of human and other creatures, cooperative learning exists wonderfully: after finding unknown virus and learning defense scheme for it, other parts of immune system learn these scheme rapidly; while they have not take part in killing offensive virus.

One of main problems in learning from others is to specify the appropriate model agent to learn from it. The model is an agent that other agents use its knowledge for their learning. In researches done on cooperative learning, the best results were obtained in learning from experts [Tan,1993]; But, it seems that non-expert agents also have some knowledge useful for others. Also, in majority of problems, no expert agents exist in the system as no solution is provided beforehand. Therefore learning from less-expert or non-expert agents becomes important in most learning problems.

When learning from other agents, the learner should specify amount of their expertness and assigns an importance weight to their knowledge. Choosing appropriate weights is one of the main problems in cooperative learning. To our knowledge, nobody has studied different solutions for this problem and compared them with each other. So, we intend to find the best weight assigning method and devise a proper way to use other agent's experiences by the learner. In this study, we introduce **Weighted Strategy-Sharing** method for cooperative learning. Also some measures to evaluate agent expertness are introduced and are applied to our system.

Related researches are reviewed in the coming section. Then, our cooperative learning strategy, called **Weighted Strategy-Sharing** is introduced. In the fourth section individual learning is explained and some expertness measures are devised. In the last sections, our methods are tested on the Hunter-Prey and Object Pushing problems.

2 Related Researches

Samuel [Samuel,1963] used the Competitive Learning algorithm to train a Checker game player. In his method, cooperator agent has the role of enemy or evaluator and tries to find weak strategies of the learner. In the Ant Colony System [Dorigo&Gambardella,1997], some ants learn to solve the Travelling Salesman Problem by non-verbal communication through the pheromones on the edges of a graph.

Imitation [Kuniyoshi et al.,1994] is one of cooperative learning methods. In imitation, the learners watch the actions of a teacher, learn them and repeat the actions in similar situations. This method does not affect the performance of the teacher [Bakker&Kuniyoshi,1996]. For example in [Kuniyoshi et al.,1994], a robot perceives a human doing a simple assembly task and learns to reproduce it again in different environments. Hayes and Demiris [Hayes&Demiris,1994] have built a robotic system in which a learner robot imitates a trainer which moves in a maze. The robot learns to escape from the maze later.

Yamaguchi and others [Yamaguchi et al.,1997] have developed a robotic imitation system to train ball-pusher robots. In this system, agents learn individually based on Reinforcement Learning, but they have the ability to imitate each other too. In their paper, three methods of imitation are used: Simple Mimetism, Conditional Mimetism, and Adaptive Mimetism.
In Simple Mimetism all agents imitate each other when they are neighbors, considering an imitation rate. In this method, Inter-Mimetism Problem is occurred when two neighbors wait to imitate each other and do nothing. This problem is solved by Conditional Mimetism in which only low performance agent (performance is measured based on sum of rewards and punishments received in *n* previous actions) imitates the other one. Adaptive Mimetism is similar to Conditional Mimetism, but imitation rate is adjusted based on the difference between performances of two neighbor robots.

Cooperation in learning also can be done when agents share their sensory data and play the role of a scout for each other [Tan,1993]. Episode Sharing can be used to communicate the (state, action, reward) triples between Reinforcement Learners [Tan,1993]. Tan showed that sharing episodes with an expert agent could improve the group learning significantly. In Collective Memory method, the learners put the learnt strategy or experienced episodes on a shared memory [Garland&Alterman,1996] or have a unique memory and update it cooperatively[Tan,1993].

A Cooperative Ensemble Learning System [Liu&Yao,1998] has been developed as a new method in neural network ensembles. In this study linear combination of outputs of concurrent learning neural networks are used as a feedback to add a new penalty term to error function of each network.

Provost and Hennessy [Provost&Hennessy,1996] developed a Cooperative Distributed Learning System which is used when the training set is huge. First, the training set is divided into *k* smaller training subsets and *k* Rule-Learning agents learn local rules from these subsets. Then, the rules are transmitted to other agents for evaluation; if the rule satisfies the evaluation criterion, it is accepted as a global rule.

Maclin and Shavlik[Maclin&Shavlik,1996] have used Advice Taking scheme to help a Connectionist Reinforcement Learner. The Learner accepts advices in the form of a simple computer program, compiles it, represents the advice in a neural network form, and adds it to its current network.

In most of the reviewed researches, cooperation is uni-directional between a pre-specified trainer and a pre-specified learner agent. In real world, Cooperative Learning is two-directional and all of agents learn something from each other (even from non-expert) agents. In Strategy Sharing Method [Tan,1993], agents of a multi-agent system learns from all of agents. The agents learn individually by Q-Learning version of Reinforcement Learning. At special times, agents gather the Q-Tables of other agents and take the average of tables as their new strategy. This system uses knowledge of all of agents but the agents do not have the ability to find good teachers. Also, this system has high communication cost because all of agents must transmit all cells of their tables. Moreover, it seems that the simple averaging of tables is non-optimal when agents have different skills and experiences. Additionally, the Q-table of agents become equal after each cooperation step which decreases the adaptability of agents to the environment changes [Yamaguchi et al., 1997]. To overcome the described difficulties, we have developed a new strategy sharing method based on expertness detection where agents assign weights to tables of other agents.

3 Weighted Strategy-Sharing Method

In Weighted Strategy-Sharing Method (Algorithm 1) it is assumed that a group of *n* homogeneous agents is learning in distinct environments, so their actions do not change the learning environment of others (and do not make the Hidden State Problem [Friedrich et al.,1996]). The agents are learning in two modes: **Individual Learning Mode** and **Cooperative Learning Mode** (lines 4 and 15 of Algorithm 1). At first, all of gents are in Individual Learning Mode. Agent *i* executes t_i learning trials, based on the One-Step Q-Learning version of Reinforcement learning (different t_is causes different experiences). Each learning trial starts from a random state and ends when the agents reach the goal. After a specified number of individual trials (which is called Cooperation Time) all agents stop Individual Learning and switch to Cooperative Learning Mode.

In Cooperative Learning Mode, each agent assigns weights to other agents with respect to their expertness. Then, it takes weighted average of Q-table of agents and uses the resulted table as new Q-table for itself.

Multiplication (*) and Summation (+) operators must be specified based on knowledge representation method. For example, in One-Step Q-Learning method, * operator is scalar matrix multiplication operator and + operator is matrix summation, etc. Expertness measures are introduced in the next section.

3-1 Individual Learning based on Reinforcement Learning

In this paper, One-Step Q-Learning is used for Individual Learning Mode. Reinforcement learning is one of the simplest and mostly used learning method that has many similarities to human experiences. In this method, the agent perceives something about the state of its environment and, based on a predefined criterion, chooses an action that it thinks is appropriate. The action changes the world's state and the agent receives a scalar "reward" or "reinforcement" indicating the goodness of the new state for the agent. After receiving reward or punishment, it updates the learnt strategy based on learning rate and other parameters.

In One-Step Q-Learning algorithm [Watkins,1989] external world is modeled as a Markov decision process with discrete time finite states. Next to each action, the agent receives a scalar "reward" or "reinforcement".

The state-action value table, Q, which estimates long-term discounted reward for each state/action pair, determines the learned policy of agent. Given a current state and available action "a_i", a Q-learning agent selects each action *"a"* with a probability(P) given by the Boltzmann distribution (line 7 of algorithm 1):

Algorithm 1-Weighted Strategy Sharing Algorithm for each agent A_i

(1) *Initialize*
(2) while not *EndOfLearning* do
(3) begin
(4)If *InIndividualLearningMode* then
(5)begin *{ Individual Learning}*
(6) x_i *:=FindCurrentState()*
(7) a_i *:= SelectAction()*
(8) *DoAction*(a_i *)*
(9) r_i *:= GetReward()*
(10)y_i *:= GoToNextState()*
(11) $V(y_i) := Max_{b \in actions} Q(y_i, b)$
(12) $Q_i^{new}(x_i, a_i) := (1 - b_i)\, Q_i^{old}(x_i, a_i) + b_i\, (r_i + g_i\, V(y_i))$
(13)e_i *:= UpdateExpertness*(r_i)
(14)end
(15)else *{Cooperative Learning}*
(16)begin
(17) for *j:= 1* to *n* do
(18)e_j *:= GetExpertness*(A_j)
(19) $Q_i^{new} := 0$
(20) for *j:= 1* to *n* do
(21) begin
(22) W_{ij} *:= ComputeWeights*(i,j,e_1...e_n)
(23) $Q_j^{old} := GetQ(A_j)$
(24) $Q_i^{new} := Q_i^{new} + W_{ij} * Q_j^{old}$
(25) end
(26) end
(27) end

$$P(a_i \mid x) = \frac{e^{Q(x,a_i)/\tau}}{\sum_{k \in actions} e^{Q(x,a_k)/\tau}}$$

where τ is the temperature parameter that adjusts the randomness of decision. The agent then executes the action (line 8), receives an immediate reward *r*(line 9), moves to the next state *y* (line 10) and updates *Q(x,a)* by this formula(line 12):

$$Q(x,a) \leftarrow (1-\beta)\, Q(x,a) + \beta\, (r + \gamma\, V(y))$$

Where b is the learning rate and g $(0 \le \gamma \le 1)$ is a discount parameter and *V(x)* is given by(line 11):

$$V(y) = \max_{b \in actions} Q(y,b)$$

As the agent searches the state space, *Q* is improved gradually.

3-2 Expertness Criteria

In Weighted Strategy-Sharing Method, weights of each agent's knowledge must be specified so that the group learning efficiency is maximized. The Expertness measuring criterion can significantly affect the learning efficiency. This can be observed in human society, in which we evaluate one's knowledge with respect to its expertness. Therefore, each agent must predict that which agent has best efficiency at the end of learning period and assign more weight to its knowledge; meaning how much it can rely on other agent's knowledge.

Different mechanisms for choosing expert agent are used. Most of the researches have pre-specified expert agent(s). For example, in Tan[Tan,1993], expert agent is predefined and is not changed during group learning. This expert agent, which is trained or programmed previously, does not learn and helps others in learning only.

In strategy sharing method [Tan,1993], expertness of agents are assumed to be equal. Nicolas Meuleau [Meuleau, 1991] have used the judgment of user for specifying expert agent. This method requires continuous supervision of human.

In [Alpaydin,1998] different but fixed expertness is assumed for each agent. Difference in expertness can be due to initial knowledge, different experiences, different learning algorithm, or different training sets. But it must be noted that, difference in expertness may change in the learning process and cannot be assumed constant.

Yamaguchi et al [Yamaguchi et al,1997] have specified the experts agent by means of their successes and failures during current *n* moves and considered the expertness criterion as algebraic sum of reinforcement signals received in that time interval. This means that more successes and fewer failures have considered as more expertness. We think that this method can be non-optimal in some situations.

For example, we know that an agent which has faced many failures has some useful knowledge to be learnt from it: It is possible that this agent does not know the ways of reaching the goal, but it knows the ways not leading to the goal and can avoid these failure situations in future. Also, the expertness of an agent at the beginning of learning process –that is not faced many failures- is less than other agents learnted a long time and naturally has confronted with more failures.

Considering discussed subjects, we have introduced six methods for comparing expertness of agents in this paper. These methods are:

1-Normal(Nrm): Algebraic sum of reinforcement signals.
2-Absolute(Abs):Sum of absolute of reinforcement signals.
3-Positive(P): Sum of positive reinforcement signals.
4-Negative(N): Sum of absolute value of negative reinforcement signals.
5-Average Move(AM): Reverse of the number of moves each agent does to reach the goal.
6-Gradient(G):Increased amount of received signals since last cooperation time.

3-3 Weight Assigning Mechanism

To decrease the communication required to exchange Q-tables, the learner can use only the Q-tables of more expert agents. Learner *i* assigns the weights based on expertness difference between itself and other expert agents regarding the following formula:

$$W_{ij} = \begin{cases} 1-\alpha_i & if \quad i = j \\ \alpha_i \dfrac{e_j - e_i}{\sum_{k=1}^{n}(e_k - e_i)} & if \quad e_j > e_i \\ 0 & otherwise \end{cases}$$

where $0 \le \alpha_i \le 1$ is **Impressibility Factor** and shows that how much each agent relies on other's knowledge. Partial weights of others knowledge are zero if they are less expert than agent *i*. If the agent *j* is more expert than agent *i*, then its weight is relative to amount of expertness difference between agent *j* and agent *i* divided by sum of other expert's differences. Substitution of this formula in weighted averaging formula (line 24 of Algorithm 1) results the following:

$$Q_i^{new} \leftarrow (1-\alpha_i) * Q_i^{old} + \alpha_i \underset{j \in Exprt(i)}{*} \sum \left(\frac{e_j - e_i}{\sum_{k=1}^{n}(e_k - e_i)} * Q_j^{old} \right)$$

$$Exprt(i) = \{ j \mid e_j > e_i \}$$

The formula shows that previous knowledge of each agent has $(1-\alpha_i)$ effect on its new knowledge and knowledge of others is used by weight α_i.

3-4 Special Cells Communication

Two mechanisms called Positive Only (PO) and Negative Only (NO) are introduced to eliminate the communication of some cells. In Positive Only, agents send only positive-value cells of their Q-tables to others and, the expertness of agents are measured by Positive criterion. In Negative Only, agents communicate only negative-value cells of their Q-tables to others and the expertness are measured by Negative criterion.

4 Simulation on Hunter-Prey Problem

"Hunter and Prey" problem [Tan,1993] is one of the classical problems to study the learning process and is a suitable testbed for comparing different learning methods. In this paper, all experiments consist of 3 hunters, which search independently a 10×10 environment to capture a prey agent. Hunters can move with speed between 0 and 1 and prey can move with speed between 0 and 0.5. Maximum speed of the hunter must be greater than maximum speed of prey since prey is intelligent (described bellow). Otherwise, hunter cannot capture the prey. The prey is captured when its distance from hunter is less than 0.5 (which is called reward field). Upon capturing the prey, the hunter receives +R reward and -P otherwise.

Each agent has a visual field in that it can locate other agent. In the studied experiments, visual field of hunter was 2 and visual field of prey was 3. Greater visual field causes the prey to escape before the hunter see it.

The states of hunter are specified with respect to the prey *(x,y)* coordinates in its local coordination frame. If hunter was not in its visual field a default state was considered. Actions of the hunter consist of rotation and changing velocity that provides movement in a two dimensional environment: $a = (V, q)$.

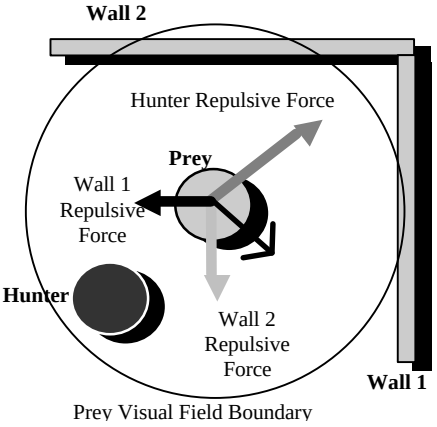

Figure 1 - an example of computing resultant force

Here, we divided distance, velocity difference, and angle difference into sections of 1 distance unit, 0.5 velocity unit and 45 degrees respectively.

For complicating the learning problem and to show the differences in efficiency of learning algorithms clearly, we have tested two simple and complex versions of the hunter-prey problem. In simple version, similar to other researches, the moving pattern of prey is irregular and random. In the complex version, the prey moves based on potential field model and escape from the hunter. We call this agent intelligent.

In the potential field model, 4 walls at each side of the environment, prey and hunter are assumed to be electropositive materials, which repulse each other. Therefore, the prey selects the path with minimum potential. Repulsive force of the hunter was considered 1.5 times of the walls. The hunter was modeled as a spot load and walls as a linear load. An example of computing resultant showed in figure 1.

In an environment with Intelligent Prey, each hunter's movements may affect the prey. So, if several numbers of hunters learn in an environment altogether, effect of an individual hunter on the prey cannot be calculated easily. So, we put only one hunter in the environment in each trail. Also for creating agents with different expertness, we have given the agents different learning times (t_is). First hunter learns 6 trials, then the second one is permitted to do 3 trials and finally, the last hunter does one learning trial. In other case the hunters have equal t_is. The total number of Individual Learning trials was 1000 and Cooperation Times were after 50 Individual Learning trials of three hunters together. Reward and punishment signals were one of 6 pairs: (10,-0.01), (10,-0.1), (10,-1), (5,-0.01), (5,-0.1), (5,-1).

In simulations, learning trials ends when the hunter captures the prey. At the beginning of each Individual Learning trial, agents are at a random. The One-Step Q-learning parameters were set to $b = 0.01$, $\gamma = 0.9$, and T=0.4. Q-table values have initialized to zero and all agents had $\alpha_i = 0.7$. Also, for trial *n*, we have measured the average number of hunter actions to capture the prey over past *n* trials.

Table 1-Improvement Percents in equal experience case.

	SA	Nrm	Abs	P	N	PO	NO	G	AM
random-prey	-0.047	-1.36	-0.141	0.375	0.235	-4.972	-23.358	0.704	2.111
intelligent-prey	-3.136	-3.339	0.936	-1.95	-0.156	-17.663	-65.096	-1.264	-0.452

Table 2- Improvement Percents in different experience case.

	SA	Nrm	Abs	P	N	PO	NO	G	AM
random-prey	-2.735	-15.307	7.582	9.933	8.301	0.845	-5.777	-7.486	-6.286
intelligent-prey	-7.572	-51.153	13.9	14.244	14.44	-1.676	-44.841	-47.49	-0.654

4-1 Equal Experiences

In figures 2 and 3, average number of moves using six mentioned Reinforcement Values are shown for individual and cooperative learning methods with the expertness criteria described above. Also, improvements of each method to Independent Learning are given in table 1.

It is clear that all of cooperative learning methods (except for Positive Only and Negative Only) have approximately the same results as Independent Learning (Ind). Positive Only and Negative Only are the worst methods. Because, these methods change the probability distribution of actions in Boltzmann selection function and make the selection probability of positive-value and negative-value actions closer. So, the probability of selecting an inefficient action raises. Negative Only is worse than Positive Only, because even significant difference between two negative-value actions makes little change in the selection probability of the actions in Booltzmann selection method. But a little difference between two positive-value actions can raise the probability of better action significantly.

4-2 Different Experiences

In figures 4 and 5, average number of moves over the 6 described Reinforcement Values are shown for each cooperation method. Also improvement percent of each method are written in table 2.

Table 2 shows that Absolute, Positive, and Negative are resulted in improvement in all cases. In random-prey case, where the number of punishments is less than intelligent-prey case, Positive criterion has the best results. But, in intelligent-prey case, Negative criterion has the best results.

Normal and Gradient have the worst results. These criteria mistake in assigning expertness to the agents. It is due to the fact that the more experienced agent has more tries and, consequently, has received more punishments. Therefore, its expertness becomes negative and less weight is assigned to its knowledge.

Average Move criterion has negative effect in random-prey case and approximately no effect in intelligent-prey case. In this criterion, the difference between expertness of agents is little, even when they have considerably different number of moves.

Negative Only has negative effect on learning in the two cases and Positive Only has approximately no effect. Simple Averaging (SA) has negative impact in two cases because of assigning equal weights to agents, which have different expertness.

5 Object Pushing

In the Object Pushing problem, simulated in this paper, two robots learn to push an object toward a target area cooperatively (figure 6). Pushing forces (F_1 and F_2) are applied horizontally at 2 points (A and B) on the object. Friction forces are fix and uniformly distributed over the environment surface. Pushing velocity is slow and inertia forces are neglected. Therefore, considering figure 6, linear (a) and angular (α) acceleration can be computed as:

$$a = \frac{F_1.\cos(q_1 - 45') + F_2.\cos(q_2 - 45') - F_S}{M}$$

$$a = \frac{F_1.r.\sin(q_1) - F_2.r.\sin(q_2)}{I}$$

where M is mass of object and I is its moment of inertia. Mechanical parameters are set to M=1(Kg), I=1, r=1(m), $m_S = 0.5$, and g=9.8(N/Kg).

To avoid Structural Credit Assignment [Claus&Boutilier,1997], we have considered a common Q-table for two robots such that actions are joint actions, consisting the first and the second robots actions: $a=(a_1,a_2)$.

States of group are specified based on coordination of mass center and angle of object relative to target: $s=(x,y,q)$. Ranges of x, y, and q are discredited to segments of 1(m), 1(m), and 45(deg), respectively. Environment size is 10×10 and since x and y can change between 10 and -10, the robots have 8×20×20=3200 states. Actions of robots are specified based on the amount and angle of forces.

Each force is applied 4 seconds and then robots follow the object until it stops. Target position is at (8,8) and initial position of object is (2,2). If the object crosses the walls the robots are positioned to initial point. Each learning trial starts from initial position and ends when the robots take the object into the target area or their number of actions exceeds from 2000 actions.

To implement cooperative learning, three groups of such systems are instantiated. Their learning trials are divided based on t_1=6, t_2=3, and t_3=1 and cooperation times occur after each 100 learning trials. Maximum number of trials is 50,000 and other parameters are $a = 0.7$, $b = 0.01$, $g = 0.9$, and T=0.4.

To define suitable reinforcement functions, we computed the average number of randomly selected moves of each group to arrive at the target (i.e. without learning) and got approximately 200 moves. Then we set rewards to 10 and adjust three cases of punishments such that sum of received punishments were

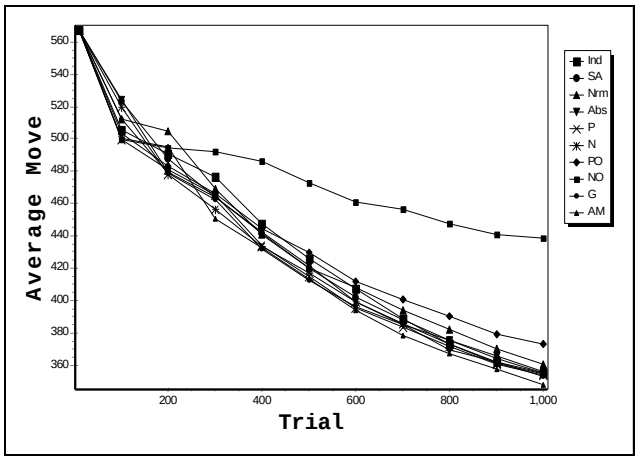

Figure 2- Average number of moves in random-prey and equal experience case.

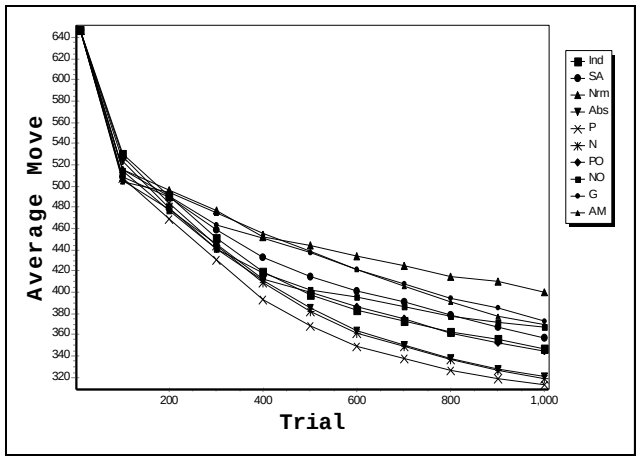

Figure 4-Average number of moves in random-prey and different experience case.

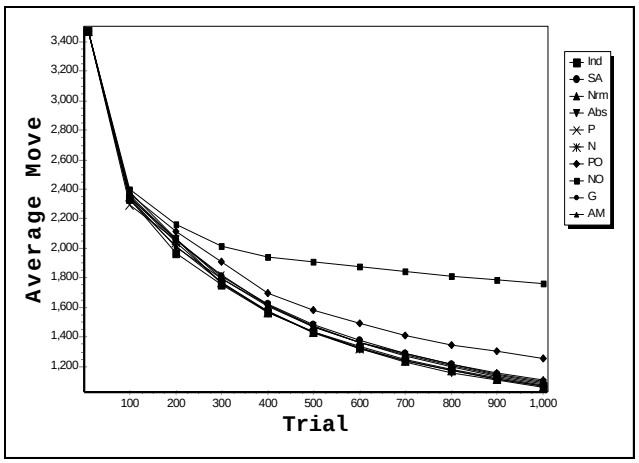

Figure 3-Average number of moves in intelligent-prey and equal experience case.

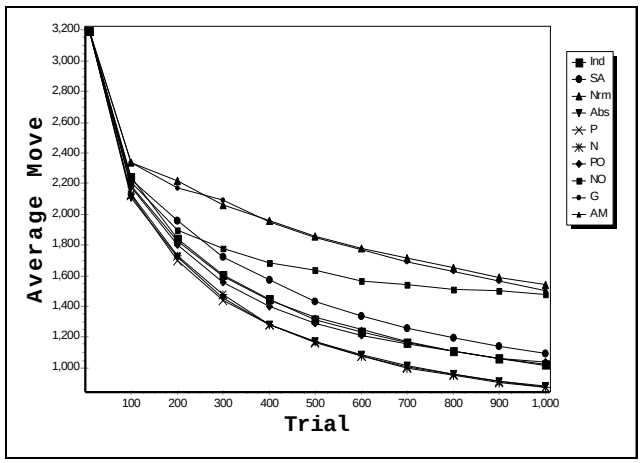

Figure 5-Average number of moves in intelligent-prey and different experience case.

approximately lower than (-0.01×200), equal to (-0.05×200), or greater than (-0.1×200) the reward (+10).

5-1 Simulation Results

A sample of the learnt path for pushing the object is shown in figure 7. The average number of moves to push the object into the target area after 50,000 learning trials is shown in table 3 for each reinforcement function. It is seen that the Simple Averaging has little positive effect on learning relative to Independent Learning. Also in all cases, Weighted Strategy Sharing based on Normal, Absolute, Positive, and Negative criteria have better results than Simple Averaging. When punishments are greater than rewards at the beginning of learning, Negative is the best criteria and Positive is the worst between the four criteria. But in the case where rewards are greater than punishments, Positive is the best criterion and Negative is the worst. Also when punishments and rewards are approximately equal, Absolute has the best results. But since any little difference in rewards and punishments significantly affects the weights in this case, the Normal criterion has the worst results. Naturally, In this case, Positive and Negative have approximately the same results.

6 Conclusion

In this paper, **Weighted Strategy-Sharing,** a cooperative learning method was introduced. Also some criteria to measure the expertness of agents were introduced and a suitable weight assigning method was presented based on expertness measuring and learning from experts. The methods were tested on Hunter-Prey and Object Pushing problems.

Results showed that Strategy-Sharing Methods had no effect (or little effect) on the learning of a multi-agent system when agents had equal experiences. When the experience of agents was different they resulted in improvement in group learning.

Simple Averaging, Gradient, Normal and Average Move had

Table 3-Average Number of moves to push the object to target area.

reward, punishment	Independent	Simple Averaging	Normal	Absolute	Positive	Negative
10,-0.01	25.0	24.8	20.6	21.4	20.6	21.5
10,-0.05	27.9	26.9	21.4	20.8	21.0	21.1
10,-0.1	27.3	26.4	20.7	21.8	22.5	20.5

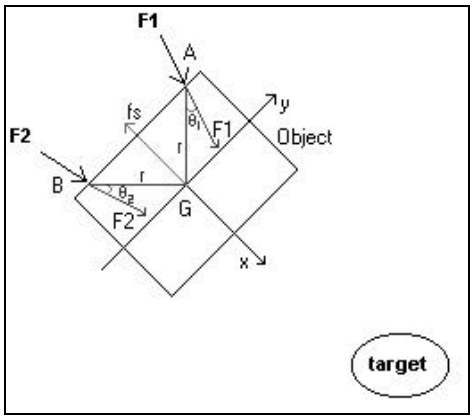

Figure 6- Object Pushing

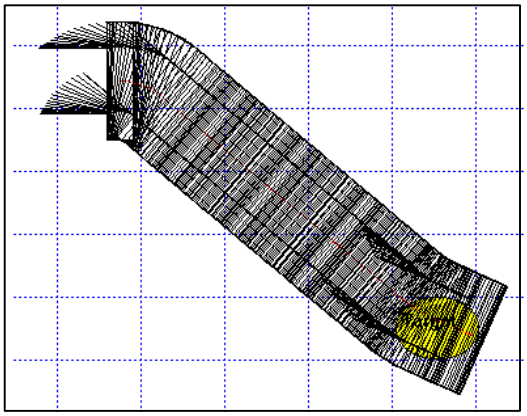

Figure 7- A sample of the learnt pushing paths.

positive effect on group learning when the learning problem was simple. In other case (e.g. Intelligent-Prey case) they had negative effects. Positive Only and Negative Only methods have completely negative effects and are not useful for cooperative learning.

The introduced criteria were sensitive to reward function of agents but Absolute criterion had the minimum sensitivity. Because, it has the properties of both Positive and Negative criteria.

Between Absolute, Normal, Positive, and Negative criteria, Positive criterion was the best method when sum of received rewards was greater than punishments in the beginning of learning. And, Negative had the worst results. In contrast, when the sum of received punishments was greater than rewards, Negative was the best and Positive was the worst. When the difference between rewards and punishments were little, Normal method had the worst results. Positive and Negative had approximately the same results, and Absolute was the best method.

References

[Alpaydin,1998] Alpaydin E., *"Techniques for Combining Multiple Learners"*, In: Alpaydin E., Proceedings of Engineering of Intelligent Systems'98 Conference, ICSC Press, Vol.2, pp.6-12, 1998.

[Bakker&Kuniyoshi,1996] Bakker P., and Kuniyoshi Y., *"Robot See, Robot Do: An Overview of Robot Imitation"*, 1996.

[Claus&Boutilier,1997] Claus C., Boutilier C., *"The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems"*, AAAI'97 workshop on Multiagent Learning, 1997.

[Dorigo&Gambardella,1997] Dorigo M., Gambardella L.M., "*Ant Colony System:A Cooperative Learning Approach to the Traveling Salesman Problem*", IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, April 1997.

[Friedrich et al.,1996] Friedrich H., Kaiser M., Rogalla O., Dillmann R., *"Learning and Communication in Multi-Agent Systems"*, Distributed Artificial Intelligence Meets Machine Learning, Lecture notes in AI, vol.1221, 1996.

[Garland&Alterman,1996] Garland A., and Alterman R., *"Multiagent Learning through Collective Memory"*, AAAISS'96, 1996.

[Hayes&Demiris, 1994] Hayes G., and Demiris J., *"A Robot controller using learning by imitation"*, In: Borkowski A., Crowley J.L.(eds.), Proceeding of the 2nd International Symposium on Intelligent Robotic Systems, pp.198-204, renoble, France: LIFTA-IMAG, 1994.

[Kuniyoshi et al., 1994] Kuniyoshi Y., Inaba M., and Inoue H., *"Learning by watching: Extracting reusable task knowledge from visual observation of human performance"*, IEEE Transaction on Robotics and Automation, Vol.10, No.6, pp.799-822, 1994.

[Liu&Yao,1998] Liu Y., Yao X., "*A Cooperative Ensemble Learning System*", Proceeding of the 1998 IEEE International Joint Conference on Neural Networks(IJCNN'98), Anchorage, USA, pp.2202-2207, May 1998.

[Maclin&Shavlic,1996] Maclin R., Shavlic J.W., *"Creating Advice-Taking Reinforcement Learners"*, Machine Learning, Vol.22, pp.251-282,1996]

[Meuleau,1991] Meuleau, N., *Simulating Co-Evolution with Mimetism*, Proc of the First European Conf. on Artificial Life(ECAL-91), pp.179-184, 1991.

[Provost&Hennessy, 1996] Provost F.J., and Hennessy D.N., *"Scaling Up:Distributed Machine Learning with Cooperation"*, AAAI'96, 1996.

[Samuel,1963] Samuel A., *"Some Studies in Machine Learning Using the Game of Checkers"*, Computer and Thought.

[Tan,1993] Tan, M., *Multi-agent reinforcement learning: independent vs. cooperative agents.* In:Machine Learning, Proceedings of the 10th International Conference, Amherst, Massachusetts, 1993.

[Watkins,1989] Christopher J.C.H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, May 1989.

[Watkins&Dayan,1992] Christopher J.C.H. Watkins and PeterDayan. *Q-Learning(technical note)*, In: Sutton R.S.(ed.), "*Machine Learning: Special issue on reinforcement learning*", vol 8, May 1992.

[Yamaguchi et al.,1997] Yamaguchi T., Tanaka Y., Yachida M., "*Speed up Reinforcement Learning between two Agents with Adaptive Mimetism*", IROS'97, pp. 594-600, 1997.