XII International PhD Workshop
OWD 2010, 23–26 October 2010

Coherent Behavior in Multiagent System Based on Reinforcement Learning

Kabysh Anton, *Brest State Technical University*
(prof. Vladimir Golovko, *Brest State Technical University*)

Abstract

This paper covers area of Collective Reinforcement Learning. We introduce and describe new simple approach to Collective Reinforcement Learning named Related Temporal Difference. This approach can supports coherence of agent's behavior in distributed and structurally complicated multi-agent system. We construct a decentralized Multi-Agent system which describes behaviors of multi-joint robot. Given experiments show, that system of local learning procedures in complex system can be much faster than learning system on the whole.

1. Introduction

More and more, machine learning is being explored as a vital component to address challenges in multi-agent systems (MAS). For example, many application domains are envisioned in which teams of software agents or robots learn to cooperate amongst each other and with human beings to achieve global objectives. Learning may also be essential in many non-cooperative domains such as economics and finance, where classical game-theoretic solutions are either infeasible or inappropriate. Teams of agents have the potential for accomplishing tasks that are beyond the capabilities of a single agent. An excellent and demanding example of multi-agent cooperation is in robot soccer.

At the same time, Multi-Agent learning (MAL) poses significant theoretical challenges, particularly in understanding how agents can learn and adapt in the presence of other agents that are simultaneously learning and adapting. This is a fertile area of research that seems ripe for progress: the numerous and significant theoretical developments of the 1990s, in fields such as Bayesian, game-theoretic, decision-theoretic, and evolutionary learning, can now be extended to more challenging multi-agent scenarios [1]. The topic of this paper is combining together Reinforcement Learning and Multi-Agent Learning to achieve new level of collective behavior of agents.

There are many principles and approaches to Multi-Agent learning [2,3]; there are some of them, important in this paper:

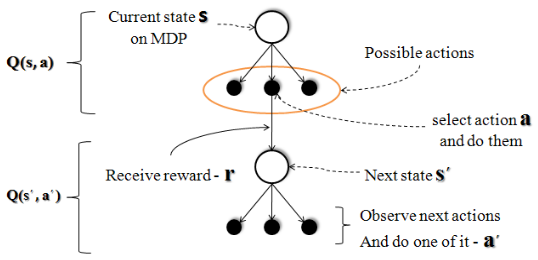
1. Some degree of decentralization of learning process.
2. Interaction between agents during learning process. Learning feedback (observer, critic, teacher, e.t.c.).
3. Involvement of agents. Interconnections and structure of Multi-Agent system must be included in learning algorithm.
4. Learning in Multi-Agent systems is on principle another kind of learning and standard techniques of single learning must be updated to use it into Multi-Agent systems.

We can use these principles as properties when we design new Collective Learning algorithm. It next sections we introduce new kind of Collective Reinforcement Learning algorithm that correspond to described principles and support's *coherence of agents behavior* into Multi-Agent Systems to produce complex, synchronized actions of agents.

2. Reinforcement Learning

Reinforcement learning is an approach to artificial intelligence that emphasizes learning by the individual from its interaction with its environment that produces optimal behavior [4]. It is often used for learning autonomous agents in unknown environment. It emerged at the intersection of dynamic programming, machine learning, biology, studies the reflexes and reactions of living organisms (reflex theory, animal cognition [5,6]. The core of all most Reinforcement Learning methods is a Temporal Difference (TD) learning [4-9]. Usually, RL used for learning autonomous agents, e.g. for robotics. Classic RL-theory works only on MDP, so it widely used for learning in game theory, e.g. TD-Gammon [10].

One iteration of RL-agent on MDP is shown at fig. 1.



Current state **s** on MDP
Q(s, a)
Possible actions
select action **a** and do them
Receive reward - **r**
Next state **s'**
Q(s', a')
Observe next actions And do one of it - **a'**

Fig. 1: One iteration of Reinforcement Learning *SARSA* algorithm (1).

Where α – learning rate, γ - discount factor.

$$\Delta Q(s,a) = \alpha[r + Q(s',a') - Q(s,a)]e(s) \qquad (1)$$

Agent does some action in particular state, goes to next state and receives reward as a feedback of recent action. During learning agent try to select the best action in some state (best action usually more rewarded in future). Learning goal is to approximate *Q*-function (1), e.g. finding true *Q*-values of *Q*-function for each *action* in every *state*.

Natural extension of standard RL algorithm is a including *eligibility traces* - are one of the basic mechanisms of reinforcement learning. Eligibility trace is a temporary records of the occurrence of an event, such as the visiting of a state or the taking of an action. At every time step (when a TD error occurs), only the eligible states or actions are updated (Fig. 2).

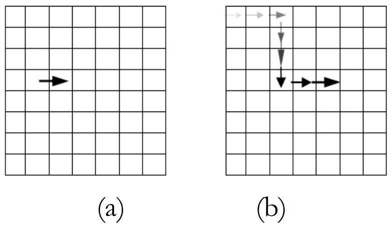


Fig. 2: Action values increase by (a) one-step SARSA, (b) by SARSA with Eligibility Trace, λ=0.9 (adopted from (A. G. Richard S. Sutton 1998)).

$$\Delta Q(s,a) = \alpha[r + Q(s',a') - Q(s,a)]e(s) \qquad (2)$$

$$e(s) = \begin{cases} \lambda\gamma e(s) & if\ s \neq s_t \\ \lambda\gamma e(s)+1 & oterwise \end{cases} \qquad (3)$$

Formula (2) called for every previously visited state if $e(s) > 0$, where $e(s)$ - is a *eligibility value,* λ - is a eligibility discount factor.

Almost any temporal-difference method, such as *Q-learning* or *SARSA*, can be combined with eligibility traces to obtain a more general method that may learn more efficiently. Its produce modified versions of algorithms used in this work *SARSA*(λ) and *Watkins-Q*(λ), *Peng-Q*(λ) and another.

There are many versions of RL algorithms for single and collective learning [11] But standard RL is limited to use in Multi-Agent RL [2,3,6]:

1. Learning time grown up with state-action space.
2. Curse of dimensionality as a legacy of dynamic programming [4];
3. Hard learning and convergence with function approximations (linear and non-linear).
4. State-action space grown exponentially depending on number of agents, and generalization techniques need to be used to avoid this problem [12].

3. Collective Reinforcement Learning

In collective learning task we must learn agents cooperatively, with other agents, including interconnections into Multi-Agent System and used rules of environment to produce expected behavior of the agents. Every agent must learn rules of environment, rules of Multi-Agent system, and their own behavior scenario to acts correctly with other agent and environment. Also, collective learning includes synchronization sequences of agent's actions to produce complex intellectual behavior. It's very important emergent effects of collective reinforcement learning.

In many articles collective reinforcement learning shown in context of game theory for founding Nash equilibrium point for group of agents. Works [1,12,13] provided generalized view to this approach, and [14] pointed, that Multi-Agent learning is a still open question.

The simplest form of collective reinforcement learning named *Joint Reinforcement Learning* [15] where the whole Multi-Agent system learned as one agent.

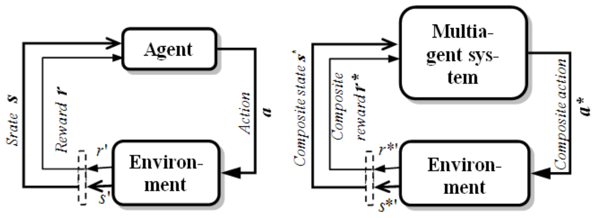


Fig. 3: Standard (left) and Joint (right) models of Collective Reinforcement Learning

Like in standard RL model, every agent in Multi-Agent system has state, and can select some action at this state. We can collect all states into one *composite* Multi-Agent system *joint state s*(t)*. Also, if some (may be all) agents in MAS produce actions at this time step, we can collect these actions into one *composite joint action a*(t)*. Environment produce *composite state s*(t+1)* and *composite reward r*(t+1)* and distribute it into MAS. After this we can learn MAS using every TD procedure in different ways.

1. *Joint MAS learning.* On Multi-Agent system level we can learn total MAS updating $Q(s^*(t), a^*(t))$
2. *Local-Joint Learning.* We can learn every agent locally updating $Q(s_i(t), a_i(t))$ for every contributed agent. To use second update rule composite reward must be divided into sub rewards for agents contributed to composite action (agent must produce action) in previous

time step *t*.

There is no principal difference between Joint RL and standard RL. Experiments with Joint-RL model have shown convergence to minimum error value with expected behavior of MAS. But, using this technique we don't avoid described limitations of RL. For Joint-RL convergence time is very slow and very sensitive to number of agents because we must search optimal policy in multidimensional state action space, where number of dimensions is equal to number of agents.

Following for state-space complexity we have problem with function approximation (but generalization potential is greater in this case). We can use different selecting technique for building composite actions to force search process, for example Genetic Algorithms with chromosome represented by composite action.

Local-Joint learning can't produce coherence structure and synchronization between agents. There is no information exchange between agents. Hence, Joint RL can be successfully applied only for simple MAL tasks, without deep synchronization and emergent effects between agents, e.g. to learn simple swarm agents.

Following for more complex Multi-Agent learning task we need to develop new collective learning techniques.

3.1 Related Temporal Difference Learning

Related TD – is new adaptation of standard TD technique for Multi-Agent system. If some problem solved cooperatively by agents, and they must learned together, so their learning is related to each other. In this case, actions from one agent may be directed to another agents (and change their states), not only to environment or himself (as in standard RL model).

Let's see to *A* and *B* - agents interconnected into one Multi-Agent system. Agent *A* actions directed not to environment, but to agent *B*. Agent *A* at state s_a execute action *a* over agent *B*, and set it into new state s_b. Agent *B* produce action *b* and execute it somewhere (on another agent, or on environment). This situation is shown at Fig. 4.

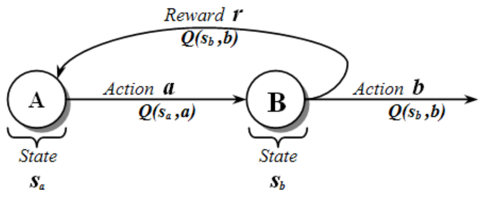


Fig. 4: Related Temporal Difference update rule

Actions *a* and *b* has their Q-*values* $Q(s_a, a)$ and $Q(s_b, b)$ respectively. Agent *B* sent to *A* feedback their *Q-value* $Q(s_b, b)$ and reward *r* as a response to action *a*. Receiving this feedback agent a can learn using standard TD technique and can update their $Q(s_a, a)$. Feedback reward r depends from agent *B*, and means its reaction to action a. Receiving this feedback agent *A* can learn using related TD update rule. Agent *A* update their *Q*-value $Q(s_a,a)$ corresponding to action a using formulas (4,5).

$$\delta_{AB} = r + \gamma Q(s_b,b) - Q(s_a,a) \qquad (4)$$

$$\Delta Q(s_a,a) = \alpha\delta_{AB} \qquad (5)$$

Formula (4) define a *temporal difference error* between agent *A* and *B*. Part of (4), $r + \gamma Q(s_b,b)$ - is a feedback from agent, to which agent *A* influence. Update rule (4) calculate TD error as measure of the inconsistency of behavior policies between for agent *A* and *B*.

Illustrated situation shows learning between two agents when state of one agent depends from actions of another (*interaction*). The main idea of related TD is that we suppose a $Q(s_b,b)$ - is a *"future" Q*-value of agent *A,* and in this case RTD is equal to TD. Feedback between agents included into update rule produces coherence of their behaviors. In this example, after learning, agent *A* will select actions that put agent B in *optimal* state.

Described learning technique extends Temporal Difference and adopts them to interactions in Multi-Agent system. This technique looks to local perspective and learn agents in multiagent system at local level including interconnections with another agents. Using related learning we can apply standard RL model locally in multiagent system. It means that we can learn agents one by one use only its local interconnection with other agents in multiagent system instead of learning system on the global level.

We have a few modifications of this technique with including eligibility traces (we call them *influence traces*) into update rule. Including eligibility traces we can reduce decentralization of learning process and propagate coherence relations more than between two agents.

3.2 Related temporal difference learning with influence trace

One of the biggest problems of collective learning – is a decentralization of learning process. How efficient to learn group of agent if they are structurally sparse far away from each other. We use term coherence to refer property of multiagent systems to be "as one organism". Such systems can easily produce synchronized actions and have many interesting properties.

Related TD with influence trace – it is a adaptation of eligibility traces to related learning described in previous section. We closely refer to idea of Eligibility traces, but change the subject storing in the trace, and way of propagation for trace. In original eligibility traces we store in memory previously visited states (see fig. 2), but in influence trace we store history (set) of agent influences to each other, as number of RTD procedures. Eligibility traces distributed in time, Influence trace – in structure (and may be in time too).

For example, let's see to more complicated and distributed example from previous chapter. Introduce one more agent **C.** This situation is shown at Fig. 5.

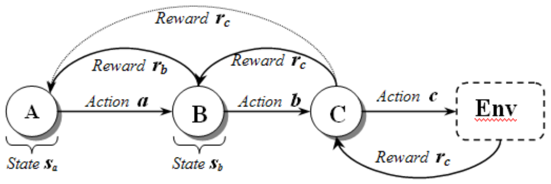


Fig. 5: RTD with influence trace for agent *C.*

We have next scenario:

1. Agent *A* acts to agent *B* with $Q(s_a, a)$. Agent *B* goes to state s_b.
2. Agent *B* acts to agent *C* with $Q(s_b, b)$. Agent *C* goes to state s_c.
3. Agent *C* acts with action *c* to environment *Env* and receive their reward.
4. Agent *B* produces feedback to agent *A* and learns it using update rule (6).

$$\Delta Q(s_a,a) = \alpha(r_b + \gamma Q(s_b,b) - Q(s_a,a)) i(d)\,|_{d=1} \quad (6)$$

5. Agent *C* produces feedback and reward to both *B* and *A* agents, and learn it using extended update rule (7-9).

$$\Delta Q(s_b,b) = \alpha(r_c + \gamma Q(s_c,c) - Q(s_b,b)) i(d)\,|_{d=1} \quad (7)$$
$$\Delta Q(s_a,a) = \alpha(r_c + \gamma Q(s_c,c) - Q(s_a,a)) i(d)\,|_{d=2} \quad (8)$$
$$i(d) = \lambda^{d-1} \quad (9)$$

State of agent *C* depends from actions and states of agents ***A*** and ***B,*** so it is forming their own *influence trace*. We introduce parameter of *influence distance* $i(d)$ that shows how far away *structurally* produced influence to this agent. Influence value is reduced with increasing influence distance between agents.

4. Experiments

To verify described RTD learning rule and compare their efficiency with Joint-RL, we test these techniques in Multi-Joined Robot (MJR) learning task. MJR model is simple decentralized model, which simulate robot arm with N-degrees of freedom, where N – is a number agents in MAS. Every segment – is an intellectual agent learned via Reinforcement Learning. The goal of experiment is to learn MJR reach some target point. This problem requires synchronization of local agent behaviors.

4.1 Model of Multi-Joined robot

MJR contains one root segment ***R***, several intermediate segments $S_1, S_2, \dots, S_m$, and one terminal segment ***T*** connected into chain from ***R*** to ***T*** (Fig. 6). Every segment, excluding terminal, can rotate at full circle (360°) all next segments. At one time step each segment, excluding terminal, can rotate all next segments at 5° to left or right, or do nothing.

First acts root segment ***R***, then first intermediate S_1, then second S_2, and so on, until S_m. Root segment can't move, can't be moved and don't change their position. Terminal segment verify reaching the target and receive actions from previous segments that change their own position.

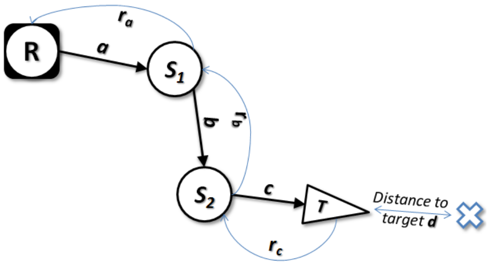


Fig. 6: Multi-Joined Robot with 4 segments *R, S_1, S_2, T.*

a,b,c **- Agent actions.** r_a, r_b, r_c **– Feedback reward corresponds to actions.**

Every segment – is an intellectual agent learned via reinforcement learning. Goal of multi-agent system is reaching a target grid. After learning MJR must reach by oneself any acceptable target cell of grid world.

Used next learning procedure (one training start):

1. MJR moved to initial position.
2. Every segment selects and executes action in order to structure of MJR. States of all next agents are changed.
3. Terminal segment calculate distance to target point.
4. If target is reached then MJR count grand-prix reward and learned. Go to 1.
5. Else, terminal segment produce feedback reward for previous agent to learn it. Feedbacks are propagated into MJR, so agents learn via RTD until root segment will be reached.
6. If simulation time is ended (1000 simulation steps) go to 1. If average RTD-Error (7) lover than limit value, then learning is over.
7. Next time step. Go to 2

4.2 Experimental Results

Learning time depends on number of segments, used algorithm and values of RL configuration parameters.

RL parameters include: α (learning rate) = 0.05~0.1; γ (discount factor) = 0.7; λ (eligibility discount factor) = 0.7~0.99, d (influence discount factor) = 0.5~0.7

Simulation of MJR behavior at first steps looks like chaotic. During learning become synchronization of behaviors between segments (successful learning) and MJR successfully can reach the target. Sometimes (5% of experiments) MJR can't synchronize at all (unsuccessful learning). In this case, behaviors of different segments compensate each other, and MJR can't successfully move in consolidate direction. It is some case of *"learning deadlock"* where robot can't get a new experience to break the lock.

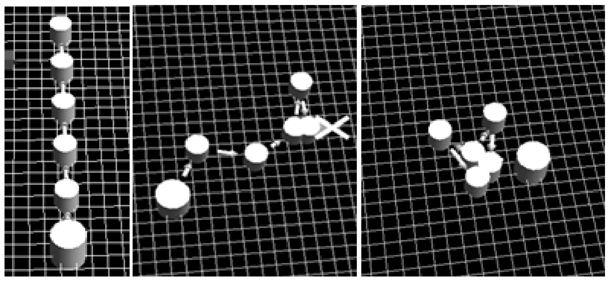


Fig. 5. Simulation of MJR in RepastJ simulation environment. (a) – initial state of MJR. (b) – successful learning, MJR reach the target. (c) – unsuccessful learning of MJR.

Quality of convergence depends from number of segments. If MJR have more than 7-10 segments then probability of convergence is much lower. Actions in the beginning of robot not synchronized with actions in the end of robot. In this case need to develop new techniques of learning for reducing complexity, or use hierarchical reinforcement learning (modular influence traces).

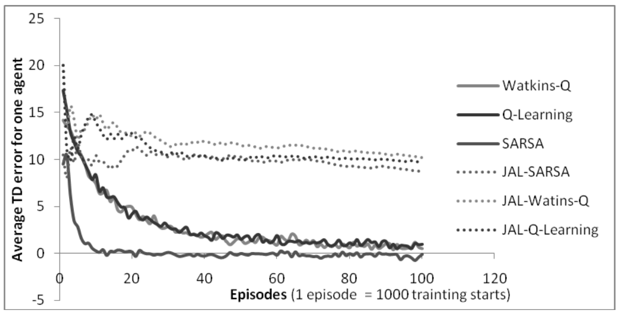


Fig. 8: Average RTD error for one agent per episode.

Fig. 8 shows efficiency of compassion Joint-RL (in legend marked as JAL) and RL algorithms under Related TD learning with influence traces. We can see experimentally that techniques using principle of local learning such as RTD and RTD convergence much faster.

Behavior policy variously changed in way of use different algorithms. RL algorithms with influence tract (SARSA(λ), Watkins-*Q*(λ)) shown more smooth behavior and better synchronization than algorithms without it (*Q*-Learning). Another unobvious result was seen in robot behavior. For algorithms with eligibility traces robot prefer rotation about a fixed root point with segment reconfiguration on new round to reach the target. Nevertheless, for *Q*-Learning (without eligibility traces) robot prefer reach the target in a straight way.

Conclusion

This work suggests new approaches to Multi-Agent Reinforcement Learning named Related Temporal Difference. This technique was designed to change standard Reinforcement Learning model in a best essential way to Multi-Agent Learning. Using RTD we can apply RL model between agents locally. We can learn agents one by one only use its local interconnection with each other, instead of learning whole system on the global level, as JAL approach. An experimental result shows faster convergence for CTD approach than for JAL in Multi-Joined Robot learning.

There are many different reward-count strategies in this MJR task. For example, we don't regulate *how* robot reaches the target. In future experiments we can calculate additional reward for "speed" or "beauty" of target reaching for robot. It is a topic of future experiments.

Bibliography

[1] Vidal, Hose M. *Fundamentals of Multiagent Systems with Net Logo Examples.* www.multiagent.com, 2009.

[2] Liviu Panait, Sean Luke. "Cooperative Multiagent Learning: The State of Art." *Autonomous Agents and Multiagent Systems, Volume 11*, 2005 : 387-434.

[3] Eduardo Alonso, Mark D'Inverno, Daniel Kudenko, Michael Luck, Jason Noble. "Learning in Multi-Agent Systems." *Science Report, Discussion.* UK's Special Interest Group on Multi-Agent Systems, 2001.

[4] Richard S. Sutton, Andrew G. Barto. "Reinforcement Learning: An Introduction." (MIT Press.) 1998.

[5] Worgotter, F. and Porr, B. "Temporal sequence learning, prediction and control - A review of different models and their relation to biological mechanisms." *Neural Computation, Volume 17*, 2005: 245-319.

[6] Dr. Florentin Woergoetter, Dr. Bernd Porr. "Reinforcement Learning ." *http://www.scholarpedia.org.* 2008. http://www.scholarpedia.org/article/Reinforcement_learning.

[7] Sutton, Richard S. "Learning to Predict by the Methods of Temporal Differences." *Machine Learning, 3*, 1988: 9-44.

[8] Barto, Dr. Andrew G. " Temporal difference learning." *Scholarpedia.org.* 2007. http://www.scholarpedia.org/article/Temporal_difference_learning.

[9] Peter Dayan, Terrence J Sejnowski. "TD(lamda) Converges with Probability 1." 1994.

[10] Tesauro, G. J. "TD-gammon, a self-teaching backgam-mon program, achieves master-level play. ." *Neural Computa-tion, 6(2), (http://www.research.ibm.com/massive/tdl.html)* , 1994: 215-219.

[11] Bab A, Brafman R I. "Multi-Agent Reinforcement Learning in Common Interest and Fixed Sum Stochastic Games: An Experimental Study." *Journal of Machine Learning Research 9*, 2008: 2635-2675.

[12] Tan, Ming. "Multiagent Reinforcement Learning. Independent vs Cooperative Agents." *Autonomous Agents and Multiagent Systems, v.10 n.3*, 2005: 273-328.

[13] Yoav Shoham, Rob Powers, Trond Grenager. "If multi-agent learning is the answer, what is the question." *Journal of Artificial Intelligence*, 2006.

[14] Stone., Peter. "Multiagent learning is not the answer. It is a question." *Artificial Intelligence, 171*, May 2007 : 402 – 405.

[15] Kabysh A., Golovko V. "Proceedings of the Tenth International Conference "Pattern Recognition and Image Processing" PRIP2009." *Collective Behavior in Multiagent Systems Based on Reinforcement Learning.* Minsk, Republic of Belarus, 2009.

Authors:

Mr. Kabysh Anton
Brest State Technical University
str. Moskovskaja 267
224017 Brest, Belarus
tel. (+375-29)7250987
fax.(+375-162) 298923
email: anton.kabysh@gmail.com