Supporting Coherence in Multiagent System: Relational Temporal Difference with Influence Trace

Surname Name

University of Applied Science
Street 000, City ZIP code, Country
mail@mail.com

Abstract

This paper introduces and compares few techniques of collective reinforcement learning. In introduction we briefly show the state-of-art of multiagent learning. We describe two learning techniques of collective reinforcement learning – *Joint-RL* and *related TD procedure* and its new modification *RTD(d)*. All this techniques can be successfully used in multiagent learning to achieve a coherence property of multiagent system. In this work we construct decentralized multiagent system which behavior describes multijointed robot, and learn every agent locally using different proposed techniques of collective reinforcement learning and compare their efficiency. 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 (Vidal 2009). Reinforcement Learning is a newer area in Machine Learning theory (A. G. Richard S. Sutton 1998). The topic of this paper is combining together Reinforcement Learning and Multiagent Learning we can achieve new level of collective behavior of agents.

There are many principles and approaches to multiagent learning (Liviu Panait 2005 , Eduardo Alonso 2001); 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 multiagent system must be included in learning algorithm.
4. Learning in multiagent systems is on principle another kind of learning and standard techniques of single learning must be updated to use it into multiagent systems.

We can use these principles as properties of multi-agent learning algorithm when we design some of it. In this paper we introduce new kind of multiagent reinforcement algorithm that correspond to described principles and support's *coherence* into multiagent systems to produce desired collective behavior of multi-agent system.

2 COLLECTIVE REINFORCEMENT LEARNING

In collective learning task we must learn agents cooperatively with other agents, including interconnections into the systems and used rules of environment to produce expected behavior of the multiagent system. Every agent must learn the rules of environment, rules of multiagent system, and their own behavior scenario and acts with these rules with other agent and environment. Also, collective learning includes synchronization of sequences of agent's actions, and coherence of multiagent structure 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 (Vidal 2009, Tan 2005, Yoav Shoham, Rob Powers, Trond Grenager 2006) provided generalized view to this approach, and (Stone. May 2007) pointed, that multiagent learning is a still open question.

2.1 Single 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 (A. G. Richard S. Sutton 1998). 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 (Worgotter 2005, Dr. Florentin Woergoetter 2008)). The core of Reinforcement Learning is method of Temporal Difference (TD) learning (Sutton 1988, Barto 2007, Peter Dayan 1994, Worgotter 2005).

RL used in robotics for learning autonomous agents. Algorithms of RL can produce optimal control (A. G. Richard S. Sutton 1991). Classic RL works only on MDP, so it widely used for learning in game theory, e.g. TD-Gammon (Tesauro 1994).

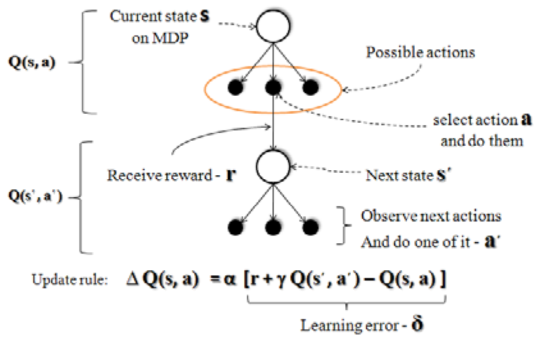


Figure 1: One iteration of Reinforcement Learning.

Reinforcement learning goal is to approximate Q-function, e.g. finding true *Q*-values of *Q*-function for each *action* in every *state*. By simple, RL is a trial-and-error learning technique. Agent try's to do some action in some state and receive reward as a feedback of recent action. During learning agent try to select the best action in some state, and best action usually more rewarded in future then another. There are existing many versions of RL algorithms. Fig 1. Shown usage of *SARSA* algorithm; another widely used RL algorithm is Q-Learning (Chris J.C.H. Watkins 2002, A. G. Richard S. Sutton 1998, C. J. Watkins 1992).

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. When a TD error occurs, only the eligible states or actions are updated (Fig. 2).

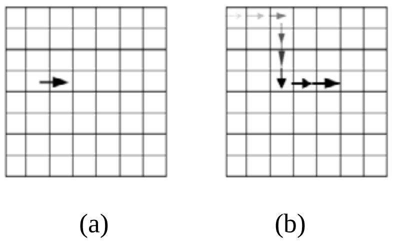


Figure 2: Action values increase by (a) one-step SARSA, (b) by SARSA with Eligibility Trace, λ=0.9 (adopted from (A. G. Richard S. Sutton 1998)).

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.

The main goal of RL is great adaptivity to learning environment. Environment may be totally unknown for agent and as a result of learning agent achieve optimal behavior on it. RL sometimes looks like agent self-adaptation to gathered environment.

Reinforcement learning – is a great learning technique, but they have some limitations if we want to use pure RL to multiagent learning (Dr. Florentin Woergoetter 2008):

1. Learning time grown up with state-action space.
2. Curse of dimensionality as a legacy of dynamic programming (A. G. Richard S. Sutton 1998);
3. Hard learning and convergence with function approximations (linear and non-linear).
4. Single agent learning as limitation of standard RL-model.
5. State-action space grown exponentially depending on number of agents, and generalization techniques need to be used to avoid this problem (Tan 2005).

2.2 Joint Reinforcement Learning

The simplest form of collective reinforcement learning named *Joint Reinforcement Learning* where on the whole multiagent system learned as one agent. Comparison of standard and Joint-RL approach to RL-model shown on Fig. 3.

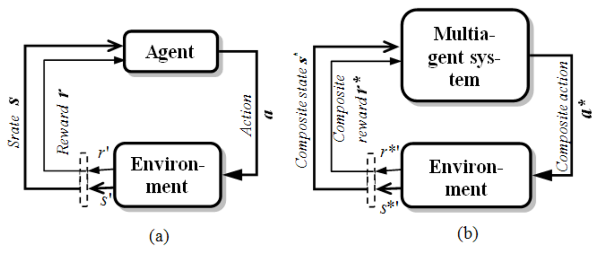


Figure 3: Standard (a) and Joint (b) models of Collective Reinforcement Learning

Like in standard RL model every agent in multiagent system has state, and can select some action at this state. We can collect all states into one *composite* multiagent 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* $a^*(t)$. In this way we can represent multiagent systems as one agent and learn it by standard RL model. Next, composite action is executed in external environment. In response to executed action, environment produce new composite state $s^*(t+1)$ and composite reward $r^*(t+1)$ for MAS. After this we can learn MAS using every TD procedure in different ways.

1. *Joint MAS learning*. On multiagent 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*.

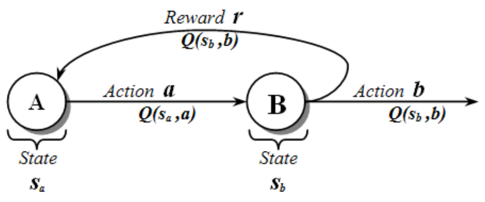
Experiments shown that Joint-RL model can produce successful learning and expected behavior of MAS, but using this model we don't avoid described limitations of RL. There is no principal difference between Joint RL and standard RL; in Joint RL we use only scale approach to standard RL. For Joint learning 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 into multiagent systems. 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 (convergences time for synchronized actions the same as for Joint MAS learning). Hence, Joint RL can be successful applied only for simple multiagent learning tasks, without deep synchronization and emergent effects. Following for more complex multiagent learning task we need to develop new collective learning techniques.

2.3 Related Temporal Difference Learning

Related TD – is new adaptation of standard TD technique for multiagent 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 multiagent system. Agent ***A*** actions directed not to environment as in standard and Joint approached, but to agent ***B***. Agent ***A*** at state $\boldsymbol{s_a}$ execute action ***a*** over agent ***B***, and set it into new state $\boldsymbol{s_b}$. Agent ***B*** produce action ***b*** and execute it somewhere (on another agent, or on environment). This situation is shown at Fig. 4.



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

Backward from next agents. TD error.

$$Q(s_a, a) = Q(s_a, a) + \alpha\delta_{AB}$$

Figure 4: Related Temporal Difference update rule

Actions ***a*** and ***b*** has their *Q-values* $\boldsymbol{Q(s_a, a)}$ and $\boldsymbol{Q(s_b, b)}$ respectively. Agent ***B*** sent to ***A*** feedback their *Q-value* $\mathbf{Q(s_b, b)}$ and reward ***r*** as a response to action ***a***. Receiving this feedback agent a can learn using standard TD technique and can update their $\boldsymbol{Q(s_a, a)}$. In feedback reward value ***r*** depends from agent B, and means reaction from it to action ***a.*** Feedback from next agent substantialize factor in related learning of two agents.

Described learning technique extends Temporal Difference method from Reinforcement Learning and adopts them to interactions in Multiagent system. Illustrated situation shows related learning between two agents when state of one agent depends from actions of another (*interaction*). Interconnections and structure of multiagent system is included RTD algorithm.

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.

2.4 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 sparse far away from each other. We use term coherence to refer property of multiagent systems to be "as one organism". Coherence multiagent systems can easily produce synchronized actions and have many interesting properties. Multiagent System may be decentralized, but learning process on it can produce coherence property. Related Temporal Difference with influence trace – is a new designed method to learning into multiagent systems. It closely referred to idea of Eligibility traces, but changes the subject of trace. In original eligibility traces we store in memory previously visited states (see fig. 2), but in influence trace we store history of agent influences in MAS as number of RTD procedures. Eligibility traces distributed in time, Influence trace – in structure.

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

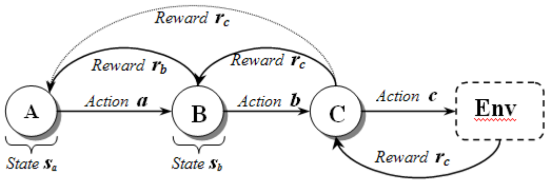


Figure 5: RTD with influence trace for agent ***C*.**

We have next scenario:

1. Agent ***A*** acts to agent ***B*** with $\boldsymbol{Q(s_a, a)}$. ***B*** goes to state $\boldsymbol{s_b}$.
2. Agent ***B*** acts to agent ***C*** with $\boldsymbol{Q(s_b, b)}$. ***C*** goes to state $\boldsymbol{s_c}$.
3. Agent ***C*** acts with action ***c*** to environment ***Env*** and receive their reward.
4. Agent ***B*** produces feedback and reward to agent ***A*** and learns it using update rule at (1).
5. Agent ***C*** produces feedback and reward to both ***B*** and ***A*** agents, and learn it using update rule (2-3).

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

$$\Delta Q(s_b,b)=\alpha(r_c+\gamma Q(s_c,c)-Q(s_b,b))i(d)\,|_{d=1} \quad (2)$$

$$\Delta Q(s_a,a)=\alpha(r_c+\gamma Q(s_c,c)-Q(s_a,a))i(d)\,|_{d=2} \quad (3)$$

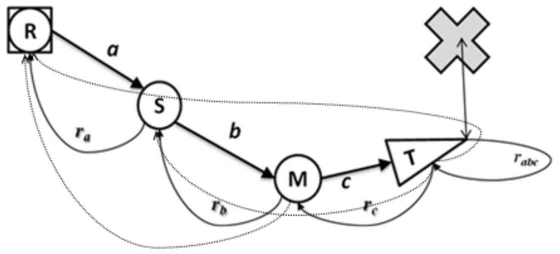
State of agent ***C*** depends from actions and states of agents ***A*** and ***B*** and this is its *influence trace*. We introduce parameter of influence distance $i(d)$ that shows how far away *structurally* produced influence to this agent. It analogous to *time* discount parameter γ or λ in eligibility trace. RTD(*d*) is a very simple intuitive recursive learning procedure.

3 EXPERIMENT

Let's use described techniques of Joint-RL and RTD(*d*) in model of multi-joined robot (MJR) and compare their efficiency in one experiment. MJR model is simple decentralized model, but learning process on it can produce coherence property – coordinated movement to the target.

3.1 Model of Multi-Joined Robot

MJR contains one root segment (**R**), several intermediate segments (**S**, **M**) and one terminal segment (**T**). At one time step each segment, excluding terminal, can rotate all next segments at 5^o in every direction or do nothing. Every action-segment can rotate at full circle (360^o) every next segment. First acts root segment, then second, and so on until terminal. Root segment can't move, can't be moved and don't change their position. Terminal segment don't produce any act, only receive actions from previous segments.



1. Root produce action ***a*** that change state of all next segments. Segment **S** produce action ***b***, and so on until terminal segment will be reached.
2. Terminal segment verify reaching the target and produce reward $\boldsymbol{r_{abc}}$ for actions ***a***, ***b***, ***c***.
3. Terminal agent calculates reward $\boldsymbol{r_c}$ for action ***c*** and starts RTD(*d*) procedure.

Figure 6: Multi-Joined Robot with influence traces and its scenario.

Every segment – is an intellectual agent learned via reinforcement learning. Agent use function approximation to build optimal policy (John N. Tsitsiklis 1997, Schneider 2006). The goal of multiagent system is reaching by terminal segment target grid cell pointed by cross. After learning MJR must reach by oneself any acceptable target cell of grid world.

3.2 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

MJR behavior in first simulations step looks like chaotic. During the learning become synchronization between segments (successful learning) and MJR successfully reach target (fig 7). Sometimes (5% of experiments) MAS can't synchronize at all. In this case different segments compensate each other and MJR can't successfully move in consolidate direction.

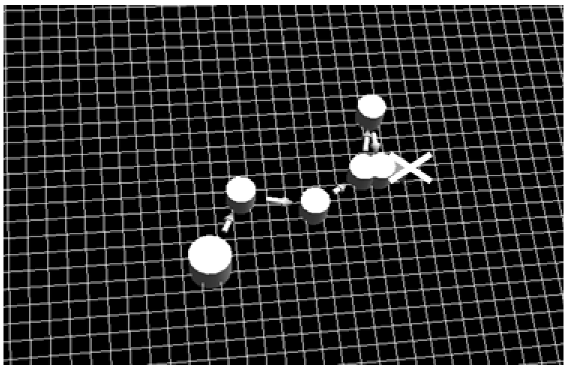


Figure 7: Modeling of Multi-Joined Robot.

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).

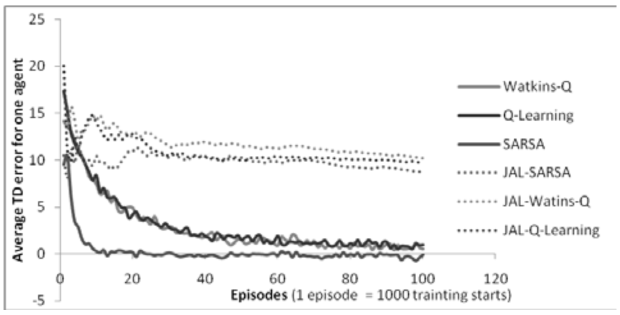


Figure 8: Average TD error for one agent per episode.

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(*d*) convergence much more faster. MJR is a decentralized learning task, so techniques with lower TD error produce more coherence and synchronization into multiagent system.

Behavior policy variously changed in way of use different algorithms. RL algorithms with influence tract (SARSA(λ,d), Watkins-*Q*(λ,d)) 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 collective reinforcement learning and looks deeply into local learning techniques in multiagent systems. We introduce few learning approaches: Joint-RL, Related Temporal Difference Learning and its modification using influence traces RTD(*d*). All of these techniques can be used in multiagent learning.

There are many different reward-count strategies in this multijointed robot task. For example, we don't regulate it what way 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.

Reference

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

Chris J.C.H. Watkins, Peter Dayan. "Reinforcement Learning." *Encyclopedia of Cognitive Science* (Wiley), 2002.

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

Eduardo Alonso, Mark d'Inverno , Daniel Kudenko , Michael Luck , and Jason Noble. *Learning in Multi-Agent Systems.* Science Report, UK's Special Interest Group on Multi-Agent Systems, 2001.

John N. Tsitsiklis, Benjamin Van Roy. "An Analysis of Temporal-Difference learning with Function Approximation." *IEEE Transactions on Automatic Control*, 1997: 674-690.

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

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

Richard S. Sutton, Andrew G. Barto, Ronal J. Williams. "Reinforcement Learning is Direct Adaptive Control." *American Control Conference.* Boston, 1991.

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

Schneider, Markus. "Reinforcement Learning with RBF-Networks." Scientific Project, University of Applied Sciences Weingarten, 2006.

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

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

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

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.

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

Watkins, Chris J.C.H. *Learning from Delayed Rewards. PhD Thesis, University of Cambridge.*

Watkins, Christopher J.C.H. "Technical Note Q-Learning." *Machine Learning 8*, 1992: 279-292.

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.

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