Advice Taking in Multiagent Reinforcement Learning

Joint work with Dr Michael Rovatsos

Alexandros-Sotiris Belesiotis
abelesiotis@gmail.com
Outline

> **Introduction**
  o Multiagent Systems
  o Multiagent Reinforcement Learning

> **Background**
  o Q-Learning
  o WoLF-PHC

> β-WoLF

> **Evaluation**
  o IPD
  o Chicken
  o Cooperation

> **Conclusion**
Introduction

The task of learning optimal policies through reinforcement learning in a multiagent environment is considerably more challenging than in the single-agent case, since the learning process also depends on the other agents' behavior.

We investigate the employment of communication of information regarding the learning problem in the form of an additional "advice" signal. And propose the $\beta$-WoLF algorithm.

$\beta$-WoLF agents are autonomous to decide upon following the advice according to

- **Rationality**: advice will only be followed if it yields payoffs higher payoffs than an individually rational strategy
- **Mutuality**: advice will only be followed if other agents are also following it
Multiagent Systems

Multiagent Systems (MAS) are the subfield of AI that aims to provide both principles for construction of complex systems involving multiple agents and mechanisms for coordination of independent agents’ behaviors [Stone & Veloso].

An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives [Wooldridge & Jennings].

An agent is considered to be intelligent if it exhibits reactivity, proactiveness and social ability [Wooldridge & Jennings].
Multiagent Systems (2)

> Useful properties of multiagent systems [Stone & Veloso]
  - Parallel computation
  - Robustness
  - Scalability
  - Simpler programming
  - Can aid in the study of intelligence

> MAS exhibit diversity regarding their environments and agents, in terms of
  - Cooperative/Competitive
  - Level of Communication
  - Open nature
  - Static/Dynamic

> Agents with hard-coded behavior will certainly fall short in complex environments. Such agents have to be adjustable and capable of learning to cope with unsuspected events and environmental changes.
Learning

> Machine Learning can be categorized
  - **Supervised Learning**
  - **Unsupervised Learning**
  - **Reinforcement Learning**

> Supervised learning deals with the problem of learning the optimal function through a series of input and output pairs, provided by some teacher or supervisor.

> Unsupervised learning deals with the problem of learning patterns in the input, without any provided relevant output.
In Reinforcement Learning (RL) agents learn through delayed rewards.

RL is a biologically inspired technique as it simulates the manner in which animals learn through signals such as pain or pleasure.

RL agents do not have to explicitly model their environment as they can learn to perform optimal actions only through the received rewards.

Reinforcement learning seems the natural choice for multiagent systems, since supervised learning is inappropriate when the optimal output for a certain input cannot be easily computed and unsupervised learning lacks the guidance reinforcement provides to the agents.
Reinforcement Learning (2)

> Reinforcement learning agents attempt to learn how to map the environment’s states to actions, in order to maximize delayed numerical rewards.

> Agent actions may cause transitions in the environment’s state. Each action the agent selects can affect both immediate and future rewards.

> RL agents have to discover the most favorable exploration/exploitation rate.

> The outcome of the agent’s learning process is a mapping from states of the environment to agent actions. This mapping is the agent’s policy.

> The rewards must not for directly revise the agents’ policies, since the agents are interested in maximizing their accumulated rewards.
Multiagent Reinforcement Learning

> The convergence of RL methods is based on the assumption that the environment is stationary.

> The learning and teaching tasks are inseparable.

> The learning task of the agent may change due to the ongoing learning process of the other agents and the effects this may have on the environment.
Outline

> Introduction
  - Multiagent Systems
  - Multiagent Reinforcement Learning

> Background
  - Q-Learning
  - WoLF-PHC

> $\beta$-WoLF

> Evaluation
  - IPD
  - Chicken
  - Cooperation

> Conclusion
Q-Learning

- Our approach is based on Q-Learning.

- Q-Learning was developed by Watkins and has been the basis of several more complicated MARL algorithms.

- A Q-learning agent maintains the value of each possible action in every state of the environment, called the Q-value.

- The agent, depending on its policy, selects the most favorable action \( a \) in its current state \( s \). Then it perceives the effects of this action in form of the new state of the environment \( s_0 \) and its reward \( r \).

- The Q-values are updated according to the formula, where \( \alpha \) is the learning rate and \( \gamma \) is the discount factor of the future rewards.

\[
Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha (r + \gamma \max_{a'} Q(s', a'))
\]

- The most commonly used policy for the Q-learning algorithm is the \( \varepsilon \)-greedy policy.
Policy Hill-Climbing

- Policy Hill-Climbing (PHC) is a rational RL algorithm, able to converge to an optimal policy against opponents using stationary strategies [Bowling & Veloso].

- However, no proof has been found that it converges against non-stationary opponents.

- Q-values are stored and updated in the same manner as in Q-learning.

- The current mixed policy is maintained, and improved in every iteration by a learning rate delta, increasing the probability of the action with the highest Q-value to be selected.
Policy Hill-Climbing Algorithm

1. Let $\alpha \in (0, 1]$ and $\delta \in (0, 1]$ be learning rates. Initialize,
   
   $Q(s, a) \leftarrow 0,$

   $\pi(s, a) \leftarrow \frac{1}{|A_i|}.$

2. Repeat,
   
   (a) From state $s$ select action $a$ according to mixed strategy $\pi(s)$ with suitable exploration.

   (b) Observing reward $r$ and next state $s'$,

   $Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a')).$

   (c) Step $\pi$ closer to the optimal policy w.r.t. $Q$,

   $\pi(s, a) \leftarrow \pi(s, a) + \Delta_{sa},$

   while constrained to a legal probability distribution,

   $\Delta = \begin{cases} 
   -\delta_{sa} & \text{if } a \neq \text{arg} \max_{a'} Q(s, a') \\
   \sum_{a' \neq a} \delta_{sa} & \text{otherwise}
   \end{cases}$

   $\delta_{sa} = \min(\pi(s, a), \frac{\delta}{|A_i|-1})$
The WoLF policy hill-climbing algorithm (WoLF-PHC) is a modification to the policy hill-climbing algorithm, employing a variable learning rate according to the “Win or Learn Fast Principle” [Bowling & Veloso].

The agent has to determine whether it is currently winning or losing.

Accordingly it selects a suitable learning parameter $\delta_i$ or $\delta_w$ (with $\delta_i > \delta_w$) for updating the learning rate $\delta$.

The WoLF-PHC is rational, since only the rate of the learning process is altered.

This modification provides more time to the other players to adapt to changes in the agent’s strategy that appear to be beneficial.

Additionally, it makes the agent more adaptive to harmful actions of the opponent.

The WoLF –PHC algorithm has been empirically proven to converge to best-response policies.
WoLF Policy Hill-Climbing

1. Let $\alpha \in (0, 1]$ and $\delta_i < \delta_{\infty} \in (0, 1]$ be learning rates. Initialize,

\[
Q(s, a) \leftarrow 0, \\
\pi(s, a) \leftarrow \frac{1}{|A_i|}, \\
C(s) \leftarrow 0.
\]

2. Repeat,

(a) Same as PHC

(b) Same as PHC

(c) Update estimate of average policy, $\bar{\pi}$,

\[
C(s) \leftarrow C(s) + 1.
\]

\[
\forall a' \in A_i; \pi(s, a') \leftarrow \bar{\pi} + \frac{1}{C(s)(\bar{\pi}(s, a') - \pi(s, a'))}.
\]

(d) Step $\pi$ closer to the optimal policy w.r.t. $Q$. Same as PHC in 3.1(c), but with

\[
\delta = \begin{cases} 
\delta_{\infty} & \text{if } \sum_{a'} \pi(s, a')Q(s, a') > \sum_{a'} \bar{\pi}(s, a')Q(s, a') \\
\delta_i & \text{otherwise}
\end{cases}
\]
Outline

> Introduction
  o Multiagent Systems
  o Multiagent Reinforcement Learning
> Background
  o Q-Learning
  o WoLF-PHC
> $\beta$-WoLF
> Evaluation
  o IPD
  o Chicken
  o Cooperation
> Conclusion
Communication may be employed as the means to enhance basic MARL algorithms.

We consider the addition of an “advice” signal to stochastic games communicating to agents information about optimal joint actions.

Our novel algorithm called $\beta$-WoLF, is based on WoLF-PHC, and enables to decide upon the degree they wish to follow the advice, based on these simple criteria:

- **Rationality**: advice will only be followed if it yields payoffs higher than an individually rational strategy
- **Mutuality**: advice will only be followed if other agents are also following it

$\beta$-WoLF essentially consists of a number of WoLF-PHC learning “modules” that learn optimal strategies for different sub-problems and a criterion for coordinating how these components are integrated by the agent to yield a single policy.
A $\beta$-WoLF agent maintains the following data structures:

> The **individual reward learner**: A WoLF-PHC module learning the policy $\pi_i(s, \alpha_i)$ that maximizes the individual rewards, using Q-table $Q(s, \alpha_i)$ and updated according to received rewards $R_i(s, a)$.

> The **collective reward learner**: This maintains the Q-table $Q'(s, \alpha_i)$ storing the values of joint actions $a \in A$ and updated according to the standard Q-learning formula.

> $n$ **individual advice learners**: One WoLF-PHC learning module per agent to simulate a pure advice-following learning process. Each one maintaining a Q-table, $V_j(s, \alpha_j)$ for $\alpha_j \in A_j$, and an advice-based policy $\rho_j(s, \alpha_j)$. The updates are made wrt the received advice $W_i$ rather than the actual reward.
\( \beta \)-WoLF (3)

> \( \beta \)-WoLF agent’s policy \( \sigma_i(s,a_i) \) is updated according to advice factor \( \beta \in [0:1] \) and an advice learning rate \( \delta \in (0:1] \) using

\[
\sigma_i(s,a_i) = (1 - \beta)\pi_i(s,a_i) + \beta \rho_i(s,a_i)
\]

> The variable \( \beta \) is depicting the extent in the agent is currently following the advice and is updated according to whether

i. the advice is rational
ii. the advice is mutually followed

using the formula

\[
\beta \leftarrow \begin{cases} 
\min \{1, \beta + \delta \beta\} & \text{if } \sum_a \prod_{j} \rho_j(s,a_j)Q'(s,a) > \sum_{a_i} \pi_i(s,a_i)Q(s,a_i) \\
\max \{0, \beta - \delta \beta\} & \text{else}
\end{cases}
\]

\text{and } d|\tilde{\sigma}_i(s) - \rho_i(s)|/dt < 0
The algorithm works in the following steps:

> With standard WoLF-PHC learning update tables $Q$ and $Q'$, adapt personal reward-based policy $\pi_i$, while updating the long-term average strategies.

> Update the $Q$-values of the advice-based WoLF-PHC learning modules, according the received advice.

> Update the overall strategy $\sigma_i$, by updating the advice coefficient $\beta$.

> To do so determine how would an increase to the advice-following behavior affect the expected accumulated rewards, based upon two criteria. The first has to do whether the proposed actions of the advice are beneficial. The second is about whether or not the other agents are following the advice.

> To determine the first we compare the expected utility for agent $I$, with all agents following the advice, to the expected utility of its individually rational strategy $\pi_i$.

> To determine the second, we check how the distance between the average opponent policy and the advice-based policy fluctuates over time.
The $\beta$-WoLF agent select its actions according to strategy $\sigma_i(s,a_i)$ or according to exploration.

- Select its following action according to its overall policy $\sigma_i(s,a_i)$ with probability $1-\varepsilon$.

- Engage in some exploration with probability $\varepsilon$.
  - If the advice is favorable and
    \[
    \sum_a \prod_j \rho_j(s, a_j)Q'(s, a) > \sum_{a_i} \pi_i(s, a_i)Q(s, a_i),
    \]
    explore randomly with probability $\varepsilon/2$, and explore wrt the advice with probability $\varepsilon/2$ for $k$ iterations.
  - Else if the advice is not considered favorable, explore randomly with probability $\varepsilon$. 
Evaluation

> We have evaluated the algorithm extensively in a number of two-player games.

> The advice was calculated by an additional module, the observer, which learns the value of the agents’ actions with respect to social welfare.

> The advice for each agent action is calculated as the “relative cooperativeness” of each player as a normalized fraction of the contribution of this player to the overall social reward of the system, compared to the action that would result in the most harmful social reward, if the other agents actions remained unchanged.
The Iterated Prisoners’ Dilemma (IPD) game is neither a fully competitive zero-sum game nor a fully cooperative team game.

This game is played by two players. Each player has to select one action to perform. This action can be either to attempt to cooperate with the other player, or to defect. Their joint actions result in the following payoffs:

<table>
<thead>
<tr>
<th>($u_1,u_2$)</th>
<th>Defect</th>
<th>Cooperate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defect</td>
<td>P(1,1)</td>
<td>T(5),S(0)</td>
</tr>
<tr>
<td>Cooperate</td>
<td>S(0),T(5)</td>
<td>R(3,3)</td>
</tr>
</tbody>
</table>

From a game theoretic perspective the dominant strategy is to defect. Therefore the only dominant strategy equilibrium (and Nash equilibrium) is mutual defection.

However, social welfare, is maximized in the Pareto efficient allocation that comes from cooperation.
Iterated Prisoners’ Dilemma

> In self-play and against opponents that followed stationary policies our agent managed to receive the intended Pareto efficient reward, discounted by the utility lost because of exploration.

> In all cases, the agent managed to end the game with cooperation, or to received the security reward.

> When possible, as with the Cooperative agent, $\beta$-WoLF managed to further increase its received reward by exploiting its opponent.

> In self-play the agent managed to learn to cooperate, and received the Pareto efficient reward.

> Against a malicious agent that initially played as a $\beta$-WoLF agent and in one point started to constantly defect to exploit our agent managed to recover and receive the finally receive the security reward.
Iterated Prisoners’ Dilemma

Average rewards against a cooperative agent
Iterated Prisoners’ Dilemma

> Average rewards against a defective agent
Iterated Prisoners’ Dilemma

- Average rewards against a tit-for-tat opponent
Iterated Prisoners’ Dilemma

> Self-play

![Graph showing iterated Prisoners' Dilemma results. The graph plots average reward against iteration, with lines representing agent rewards, opponent reward, and total reward.]
Iterated Prisoners’ Dilemma

Self-play, probability of cooperation
Chicken

The chicken game is based on the situation in which the two players drive their cars in opposite direction on the same road. Each one can drive ahead or swerve. The payoff matrix for the game is

<table>
<thead>
<tr>
<th>((u_1, u_2))</th>
<th>Swerve</th>
<th>Drive Ahead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swerve</td>
<td>3,3</td>
<td>4,2</td>
</tr>
<tr>
<td>Drive Ahead</td>
<td>4,2</td>
<td>1,1</td>
</tr>
</tbody>
</table>

The significance of the game is that if a player attempts to receive the higher reward, will risk to actually receive the lowest.

Our agent managed to maintain the security reward in all cases, in such a competitive game.
Coordination

> The coordination game is based on the situation in which the two players receive the higher outcome if they manage to coordinate and play the same action.

> In this game the agents received a payoff of 10, for selecting the same action, and a payoff of zero, for selecting different actions.

> Our agent managed to follow the advice and learn to coordinate in self-play, thus receiving the higher reward.
Outline

> **Introduction**
  o Multiagent Systems
  o Multiagent Reinforcement Learning

> **Background**
  o Q-Learning
  o WoLF-PHC

> **β-WoLF**

> **Evaluation**
  o IPD
  o Chicken
  o Cooperation

> **Conclusion**
Conclusion

> We presented $\beta$-WoLF, a MARL algorithm that takes into consideration an additional advice signal.

> The algorithm increases the computational complexity of the system.

> Additionally, it requires information about the agents’ joint actions to be communicated to all agents.

> However empirical data has shown that the advice enables optimally coordinated behavior, even in games that most current MARL algorithms fail.

> Furthermore, due to its modular design it does not require complex learning modules.

> Finally, even in the cases that the communicated advice proved to be useless, the agent completely disregarded it, while maintaining its autonomy.
More Information

- M. Rovatsos, A. Belesiotis. Advice Taking in Multiagent Reinforcement Learning (poster), Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-07), Honolulu, Hawaii, USA, 2007


Advice Taking in Multiagent Reinforcement Learning

Joint work with Dr Michael Rovatsos

Alexandros-Sotiris Belesiotis
abelesiotis@gmail.com