Policy hill-climbing WoLF algorithm for rational and convergent learning in stochastic games

Ardavan Khalij [1], Kevin Sam[2] and Nicolas Rowies[3]

[1]Vrije Universiteit Brussel, Student number: 0585706
[2]Vrije Universiteit Brussel, Student number: 0558882
[3]Vrije Universiteit Brussel, Student number: 0547647

Abstract

This paper will try to re-implement the policy hill-climbing algorithm and the WoLF policy hill-climbing algorithm based on Bowling and Veloso (2001) and test them on four different games. The WoLF policy hill-climbing algorithm has two different learning rates, switching its learning rate based on the agent's state. We also introduced an enhancement for the WoLF policy hill-climbing. In this enhancement, we introduced a new way of calculating the learning rate based on the state of the agent in the game that helps the algorithm converge faster while it is rational. This paper aims to check and reproduce the results of Bowling and Veloso (2001) and also show that the introduced algorithm converges faster to the Nash equilibrium state in the game.

Introduction

In this study, the algorithm "policy hill-climbing" (PHC) and its suggested improved version, "PHC win-or-learn-fast" (PHC-WoLF), will be compared. We evaluate these algorithms' performances using a variety of games. We begin by testing the two algorithms on simple games such as Matching-Pennies, in which two players bet on the outcome of a coin flip, and Rock-Paper-Scissors, a game with three possible actions played by two players where each action can beat another one. Next, we will move on to more complex games, starting with GridWorld, a two-player game set in a 9-square environment where two players must race to reach a target cell. This is a general-sum game which allows for both cooperative and defecting behaviour. We will observe how the two algorithms perform in this setting. Finally, we will investigate the "Soccer" game, described in the Littman (1994) paper, which is a zero-sum game in which two players compete to score in their opponent's goal. We present a detailed analysis of each game, including their implementation, reward functions, and expected results. We will compare the performance of both the PHC and PHC-WoLF algorithms for each game and discuss the differences in their performances. Our goal is to replicate the results of paper Bowling and Veloso (2001) and explore the potential of further improvement based on the PHC-WoLF algorithm by developing our variation.

Methods

Stochastic Games

Stochastic games Shapley (1953) are tuples $(n, S, A_{1...n}, T, R_{1...n})$. The n represents the number of agents in the game, S S is the set of all the possible states an agent can be in, A_i is the set of actions that the agent i can take. We call A the joint action, i.e. the actions of all the agents together. A is defined as $A_1 \times ... \times A_n$. T is a transition function representing the transition from one state to another. It is defined as $S \times A \times S \rightarrow [0, 1]$. The reward function R_i is defined as $S \times A \rightarrow R$. In this setting, each agent has its own unique reward function.

The joint actions define each game iteration's rewards and the next state. In this paper, the goal is to analyze the WoLF algorithm on different games. We tried the algorithm on different games namely: matching pennies, rock-paper-scissors, Gridworld, and the Soccer game. The games were designed as stochastic games. A game where two agents have opposed goals is called a zero-sum game Littman (1994). In this type of game, the reward of one agent is the opposite reward of the other agent. When agents do not have opposing interests, their rewards do not have to be opposite. Such games are called general-sum games Hu et al. (1998). A zero-sum game is a particular type of general-sum game. Three of the four games tested are zero-sum games, namely: matching pennies, rock-paper-scissors, and the Soccer game.

Policy Hill-Climbing

A form of optimization procedure called policy hill-climbing is used to identify the best policy in a reinforcement learning situation. Policy hill-climbing (PHC) is a straightforward way to extend Q-learning to play mixed tactics, as demonstrated in figure 1 from Bowling and Veloso (2001). It functions by beginning with a basic policy and incrementally making minor adjustments to the policy to enhance the performance of the policy. The algorithm assesses the effectiveness of the existing policy at each stage and decides whether or not the adjustments made to the policy were an improvement. If the adjustments were an improvement,

the system preserves them and keeps making adjustments. If the adjustments don't improve, the algorithm throws them out and begins again with the old policy. The procedure continues until further performance gains are not possible, at which time the algorithm terminates, and the final policy is returned as the solution.

1. Let α and δ be learning rates. Initialize,
$$Q(s,a) \leftarrow 0, \qquad \pi(s,a) \leftarrow \frac{1}{|\mathcal{A}_i|}.$$
2. Repeat,
   (a) From state s select action a with probability $\pi(s,a)$ with some exploration.
   (b) Observing reward r and next state s',
   $$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a')\right).$$
   (c) Update $\pi(s,a)$ and constrain it to a legal probability distribution,
   $$\pi(s,a) \leftarrow \pi(s,a) + \begin{cases} \delta & \text{if } a = \text{argmax}_{a'} Q(s,a') \\ \frac{-\delta}{|A_i|-1} & \text{otherwise} \end{cases}.$$

Figure 1: Policy Hill Climbing algorithm for player i

WoLF Policy Hill-Climbing

WoLF Policy Hill-Climbing or WoLF-PHC has been introduced in Bowling and Veloso (2001). This algorithm is one of the variants of the PHC itself, and it is a modification of it. The main difference is that the learning rate is variable for the winning and losing states. The core concept is to change the algorithm's learning rate to promote convergence while maintaining rationality. So they decided to use the WoLF approach. This means that the method is to learn quickly while losing and slowly while winning, so the learning rate is bigger when the algorithm is in the losing state. The algorithm decides whether it is a losing state or the winning state by comparing the current policy's expected payoff with the average policy over time.

This approach helps the losing player adapt faster, so the convergence improves. The changes for the WoLF policy hill-climbing algorithm are illustrated in figure 2 from Bowling and Veloso (2001).

WoLF Policy Hill-Climbing+

This algorithm is based on the policy hill-climbing WoLF with some changes and improvements. There are two learning rates for when the algorithm is in a winning state and one for when the algorithm is in a losing state. But we decided to have a different approach to the value of the learning rate. In this approach, this value can be any number between the learning rate of winning and the learning rate of losing instead of only being limited to two numbers. So the learning rate will be calculated based on a certain threshold that can

1. Let $\alpha, \delta_l > \delta_w$ be learning rates. Initialize,
$$Q(s,a) \leftarrow 0, \qquad \pi(s,a) \leftarrow \frac{1}{|\mathcal{A}_i|}, \qquad C(s) \leftarrow 0.$$
2. Repeat,
(a,b) Same as PHC in Table 1
   (c) Update estimate of average policy, $\bar{\pi}$,
   $$\begin{aligned} C(s) &\leftarrow C(s)+1 \\ \forall a' \in \mathcal{A}_i \quad \bar{\pi}(s,a') &\leftarrow \bar{\pi}(s,a') + \\ &\quad \tfrac{1}{C(s)}\left(\pi(s,a') - \bar{\pi}(s,a')\right). \end{aligned}$$
   (d) Update $\pi(s,a)$ and constrain it to a legal probability distribution,
   $$\pi(s,a) \leftarrow \pi(s,a) + \begin{cases} \delta & \text{if } a = \text{argmax}_{a'} Q(s,a') \\ \frac{-\delta}{|A_i|-1} & \text{otherwise} \end{cases},$$
   where,
   $$\delta = \begin{cases} \delta_w & \text{if } \sum_a \pi(s,a)Q(s,a) > \sum_a \bar{\pi}(s,a)Q(s,a) \\ \delta_l & \text{otherwise} \end{cases}.$$

Figure 2: WoLF Policy Hill Climbing algorithm for player i

be chosen based on the game. Let's call this threshold T, and then the learning rate will be calculated based on figure 3.

If $\sum_a \pi(s,a)Q(s,a)$ be X_1 and $\sum_a \bar{\pi}(s,a)Q(s,a)$ be X_2:

If $X_1 - X_2 > T$ then $\delta = \delta_w$

If $X_1 - X_2 \leq T$ *and* $X_1 - X_2 > 0$ then $\delta = \delta_w + \frac{(T-(X_1-X2))(\delta_l-\delta_w)}{2T}$

If $X_1 = X_2$ then $\delta = \frac{\delta_w+\delta_l}{2}$

If $X_1 - X_2 \geq T$ *and* $X_1 - X_2 < 0$ then $\delta = \delta_l - \frac{(T+(X_1-X2))(\delta_l-\delta_w)}{2T}$

If $X_1 - X_2 \geq T$ then $\delta = \delta_l$

Figure 3: WoLF Policy Hill Climbing+ algorithm learning rate calculation

Matrix Games

We tested the algorithm on two different matrix games, namely: matching pennies and rock-paper-scissors. These games can be represented using what we call a reward matrix or a payoff matrix. The payoff matrix of the two games were defined as follows:

$$a = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{bmatrix}$$

In these two games, the agents' rewards are determined by their joint action, and here we are not considering any state. In this type of game, we can get the reward of a particular agent by viewing either the row or the column player as the action of the current agent and the other action being the other agent's action.

These games are implemented directly with the PHC and Wolf-PHC algorithm. The PHC and WoLF-PHC algorithms are designed for problems that have states and matching pennies and rock-paper-scissors don't have any meaningful states, so we used a strategy to change the states. So an agent will continue the same strategy if it has a reward of 0 or more than zero, and it will change state otherwise.

Gridworld

The Gridworld game is a game in which two agents start at a certain position. The goal of the two agents is to reach a certain position in the grid. The action space of each agent is $A_i = [North, South, East, West]$ and the game has been implemented as a gym environment. Brockman et al. (2016). We reproduced the version of the game introduced in the paper Bowling and Veloso (2001). In this version of the game, there are two walls that the agents have to traverse when going North from their starting position. When trying to go through these walls, the agent has a 50 percent chance of succeeding. In case it does not succeed, the agent stays in its original position. Another rule is that the two agents can not move to the same place at the same time. If the agents are trying to move to the same position, both of their actions fail and they stay in their original position. To avoid that, agents would wander forever before going to the goal state, agents receive a negative reward for moving, and the reward for going into the goal state is 0. Clearly, the optimal solution involves some cooperation; otherwise, the agents could block themselves infinitely.

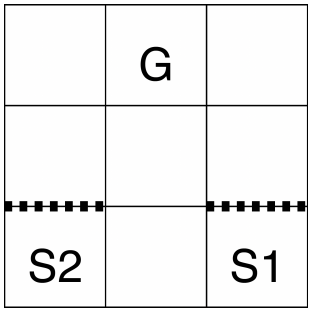

Figure 4: Starting state of the Gridworld

Soccer

The Soccer game introduced in Littman (1994) is a game in which two agents play soccer. This game was also implemented as a gym environment. The two agents start on their side of the field each. The action space of each agent is $A_i = [Stay, North, South, East, West]$ Their position at the start of the game is random. One of the agents is chosen randomly to have the ball at first. An agent loses the ball by trying to move to a place where there is another agent. When this happens, the stationary agent gets the ball. The agents' goal is to bring the ball to the goal on the opposite side of the field. When an agent scores a goal, he gets a reward of 10, and the other agent receives a reward of -10. In this game, there is no penalty for wandering. The game is played on a 5x4 matrix, with four additional positions for the goals (a goal contains two positions).

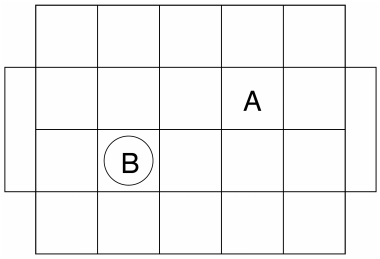

Figure 5: Example starting position of the Soccer game

Results

In this section we discuss the results of each game, we do this with the help of illustrations done by graphical representations. We employ a variety of plot types to demonstrate each game's performance clearly. Before starting to discuss the results of each game it is important to keep in mind that the Bowling and Veloso (2001) paper does not contain an appendix or specify the parameters in the paper for the majority of the games, so there might be a few minor variations between the results we discuss and those in the paper. Nonetheless, our findings are generally consistent with those in the paper.

Matching Pennies

We compared the PHC and WoLF-PHC by plotting the possibility of choosing head by the agent i. The initial policy was to set the odds of playing head higher than the odds of playing tails to see if the algorithm could converge on the expectations. Those expectations are that both actions have equal probability. Figure 6 illustrates this game in the PHC algorithm, and it is clearly visible that the policies do not converge to a certain number.

Figure 7 shows the probability that agent i chooses heads in the WoLF-PHC algorithm and the convergence is clear in the graph. You can see that there are small wave movements like with the PHC algorithm at the beginning. But just after

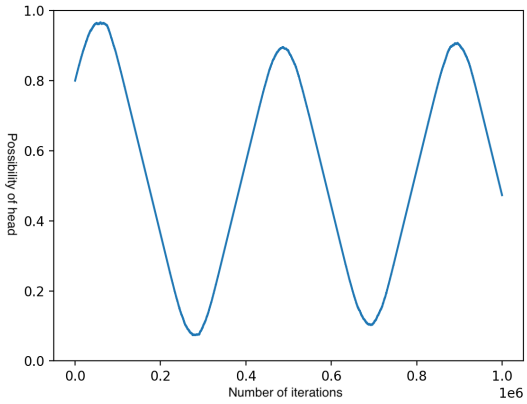

Figure 6: Matching-Pennies game PHC algorithm

two or three waves, the WoLF-PHC algorithm converges to the desired policy.

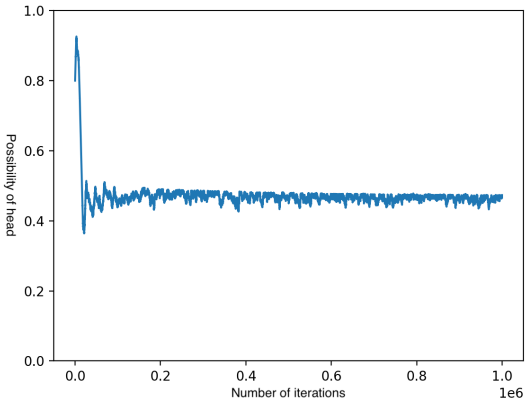

Figure 7: Matching-Pennies game PHC-WoLF algorithm

Rock Paper Scissors

We compare two actions in the game of rock-paper-scissors to see if there is any improvement. Since it is known that each of the three actions in this game has an equal chance of succeeding, a perfect policy would assign each action an equal probability. We initially give the actions in our policy unequal probabilities to test how well the algorithms work. In doing so, we will be able to see if the policy tends toward an equal distribution. For simplicity of representation, we only consider the ”paper” action and the ”rock” action in the following graphs. For the initial policy, playing paper is set to 25%, playing rock is also set to 25% in probability, and playing ”scissors” has 50% probability. More details on the parameters chosen to run the algorithms and the initial policies can be found in the appendix.

In the following graphs, the first 200 points are in red to make it easier to see the direction in which the agent is learning. Now when we observe the graphs, we can see from the results in 8 that the PHC algorithm does not result in a stable policy because the agent continuously modifies it.

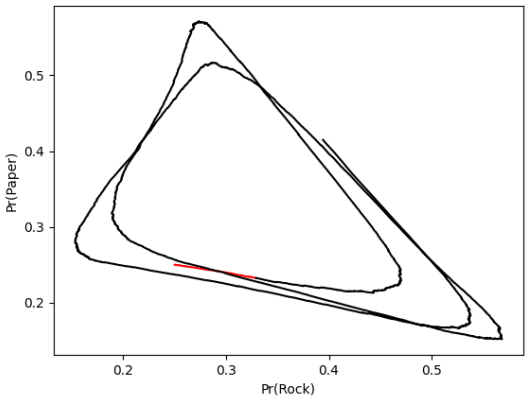

Figure 8: Rock-Paper-Scissors game PHC algorithm

Figure 9, on the other hand, demonstrates that the agent rapidly converges to a stable policy. In addition, based on the first 200 episodes and how it stretches, the agent in figure 9 seems to pick things up more quickly than the PHC algorithm.

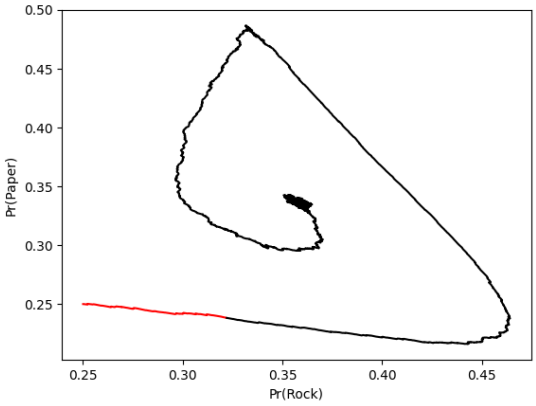

Figure 9: Rock-Paper-Scissors game PHC-WoLF algorithm

GridWorld

We observe how the agents act in their initial positions in the GridWorld game. There are initially two logical options available to them. They can decide to take the action to move on their side (East or West depending on the agent), but if the opposing agent does the same, no one moves and both suffer a penalty of one. Alternately, they have a 50/50 chance of

success if they head north, but doing so increases the likelihood that the other agent will go East/West and outperform the agent. The objective is to observe how the agents attempt to optimize their behaviour by taking into account the behaviour of the other agent. For the initial policy, we choose the same policy for both agents, which will initially favour the action towards the north and all other action probabilities will be distributed equally.

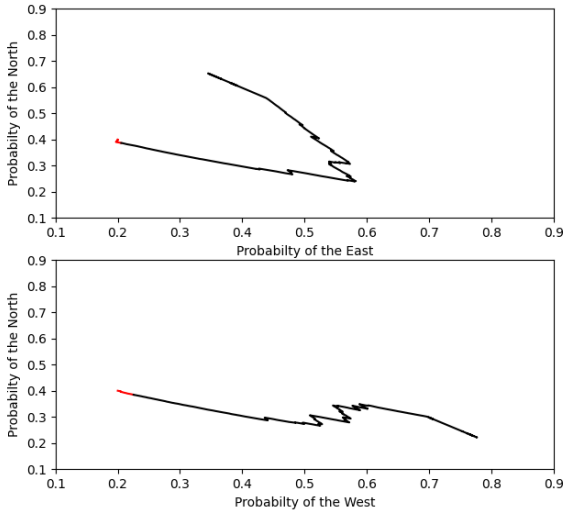

Figure 10: Agent 1 choosing to go North, agent 2 choosing to go West

The outcomes demonstrated in figure 10 and 11 show that the agents alternately cede the side position. Agent one seems to move in that direction occasionally and something chooses to go up. This appears to be the case because when one agent starts to take advantage of the side action, the other will logically choose the north action to prevent losing more. So it all depends on which agent is going to choose to side action first, and the other will then adapt.

Soccer

In the results, first of all, we note that we were unable to carry out 50 runs due to a lack of computational availability. So the results in our graphs, although similar, are to be taken more lightly than Bowling and Veloso (2001). Besides that, our conclusion is almost the same, the Minimax-Q algorithm results were directly taken from the paper Littman (1994). PHC(W) is the PHC algorithm run with $\delta = \delta_{win}$, similarly, PHC(L) is the PHC algorithm run with $\delta = \delta_{loss}$. We ran each algorithm for one million iterations against itself, and then after that, we stopped the learning for one of the agents and we continued the learning of the other agent. When we wrote 1X, it means that the algorithm was run for

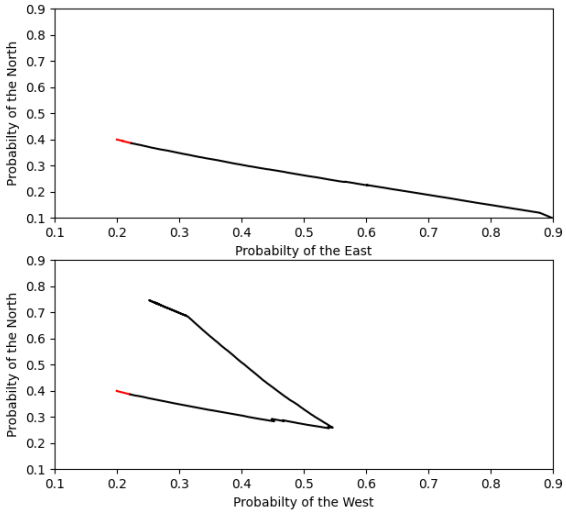

Figure 11: Agent 1 choosing to go East, agent 2 choosing to go North

125 000 000 iterations and we stopped the learning of one agent at 1 million iterations. When we wrote 2X it means that the algorithm was run for 2 million iterations and the learning of one agent was stopped after 1 million iterations. We see very quickly in the graph 12 that the performances of the algorithms WoLF and WoLF 2X are largely superior to that of PHC(W) and PHC(L). We, therefore, conclude the same as in the paper Bowling and Veloso (2001) that the WoLF algorithm is performing better. We note a big difference between our results and those of paper Bowling and Veloso (2001) in the standard deviations. But again, we have strong doubts that this is related to the fact that we made far fewer runs than the paper. Furthermore, we also note that our enhanced version of the PHC-WOLF algorithm, which we called PHC-WoLF+, is performing better than its original counterpart. We can even see that PHC-WoLF+ 1X, outperforms the PHC-WoLF 2X, which means that the PHC-WOLF+ converges to a solution faster than the PHC-WoLF. We can also see on the plot that the PHC-WoLF+ 2X has better results than the PHC-WoLF 2X, which shows that the algorithm still continues to improve, with more training. Furthermore, the standard deviation of the PHC-WoLF+ algorithm is smaller than the standard deviation of the PHC-WoLF algorithm. This shows us that the PHC-WoLF+ algorithm has less variability in its leaning curve, than the PHC-WoLF algorithm.

Discussion

In this paper, we aimed to reproduce the results of the paper Bowling and Veloso (2001). We have obtained similar

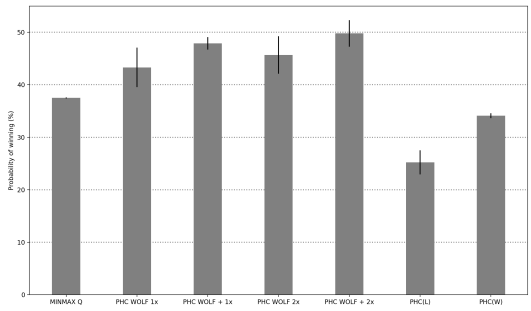

Figure 12: Algorithms performance comparison on the soccer game

results using the PHC and the PHC-WoLF algorithm. The PHC-WoLF algorithm is an algorithm that uses an interesting idea, namely, having different learning rates for winning and for losing. The idea is that the agents should learn slowly while winning and learn fast when losing. Although an interesting idea, having just one learning for winning and another for losing is not the optimal solution. In our version of the algorithm, the learning rate is truly variable, and the δ_{loss} and δ_{win} are used as limits for the actual learning rate that is used at any particular iteration. The idea remains the same: agents should learn fast when losing and slow when winning. However, sometimes the agent may win, but in a very slow and thus inefficient manner, and the opposite is also possible. It is possible that the agent loses but that its strategy was a good strategy to win. By using truly variable learning rates, we tried to implement this idea in our algorithm, and we see that this improves the agent's learning.

References

Bowling, M. and Veloso, M. (2001). Rational and convergent learning in stochastic games. In *International joint conference on artificial intelligence*, volume 17, pages 1021–1026. Lawrence Erlbaum Associates Ltd.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Hu, J., Wellman, M. P., et al. (1998). Multiagent reinforcement learning: theoretical framework and an algorithm. In *ICML*, volume 98, pages 242–250.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier.

Shapley, L. S. (1953). Stochastic games. *Proceedings of the national academy of sciences*, 39(10):1095–1100.

Matching pennies

Parameters

Learning rate (alpha)	0.01
Delta	0.00001
Delta win	0.00005
Delta lose	0.0001
Gamma	0.99
Average every	100

Initial policy

Heads	0.8
Tails	0.2

Rock Paper Scissors

Parameters

Learning rate (alpha)	0.01
Delta	0.00001
Delta win	0.0001
Delta lose	0.0002
Gamma	0.99
Average every	100

Initial policy

Rock	0.25
Paper	0.25
Scissors	0.5

GridWorld

Parameters

Learning rate (alpha)	0.01
Delta	0.00001
Delta win	0.0001
Delta lose	0.0004
Gamma	0.99
Average every	100

Initial policy

Agent 1

North	0.4
East	0.2
South	0.2
West	0.2

Agent 2

North	0.4
East	0.2
South	0.2
West	0.2

Soccer

Parameters

Learning rate (alpha)	0.01
Delta	0.00001
Delta win	0.000001
Delta lose	0.000004
Gamma	0.99
Average every	100
High limit for delta calculation in WoLF+	0.03
Low limit for delta calculation in WoLF+	-0.03

Initial policy

Agent 1

North	0.2
East	0.2
South	0.2
West	0.2
Stay	0.2

Agent 2

North	0.2
East	0.2
South	0.2
West	0.2
Stay	0.2