Multi-agent system with Policy Gradient Reinforcement Learning for RoboCup Soccer Simulator

ALEXANDER GOMEZ, VIKTOR GAVELLI

Degree Project in Computer Science, DD143X
Supervisor: Christian Smith
Examiner: Örjan Ekeberg

CSC, KTH, 2014-04-29
Abstract

The RoboCup Soccer Simulator is a multi-agent soccer simulator used in competitions to simulate soccer playing robots. These competitions are mainly held to promote robotics and AI research by providing a cheap and accessible way to program robot-like agents. In this report a learning multi-agent soccer team is implemented, described and tested. Policy Gradient Reinforcement Learning (PGRL) is used to train and alter the strategical decision making of the agents.

The results show that PGRL improves the performance of the learning team. But when the gap in performance between the learning team and the opponent is big the results were inconclusive.
Sammanfattning


Resultaten visar att PGRL förbättrar lagets prestanda, men när lagets prestanda skiljer sig avsevärt från motståndarens blir resultatet ofullständigt.
Contents

1 Introduction
   1.1 Question Formulation

2 Background
   2.1 RoboCup Soccer Simulator
   2.2 Controls
   2.3 Rules
   2.4 The trainer

3 Method
   3.1 Formation
   3.2 Actions available to player in possession of the ball
      3.2.1 Dribble
      3.2.2 Pass
      3.2.3 Shoot on goal
   3.3 Actions for player not in possession of the ball
      3.3.1 Go to ball
      3.3.2 Hold formation
      3.3.3 Cover opponent
      3.3.4 Get open for pass
   3.4 Goalkeeper specific actions
      3.4.1 Catch ball
      3.4.2 Cover goal
   3.5 Decision making
      3.5.1 Policy
      3.5.2 PGRL
   3.6 Training
   3.7 Experiments

4 Results
   4.1 The team against its non-learning equivalent
      4.1.1 Same starting parameters
Introduction

Strategical decision making of a multi-agent soccer team is a challenging multidimensional decision problem. An efficient solution requires that the agents cooperate and coordinate their actions while taking positions and velocities of all objects on the field into consideration. This is further complicated by the rules of the game and the different states of play, such as free kicks and kick-offs. A popular platform for simulated robot soccer is the RoboCup Soccer Simulator (RCSS) [2].

RoboCup is a robotics competition where teams of robots play football against each other. The aim of the RoboCup is to promote artificial intelligence and robotics research through publicity. Research applied to RoboCup football teams could also be used in other multi-agent applications where teamwork is needed, for example urban search and rescue robotics.

Implementing an efficient team for the RCSS consists of many subproblems. The team has to be good at many things, individual actions such as passing, intercepting the ball, positioning and shooting for a goal; but also strategizing, making adequate strategical decisions about when to pass or shoot at the goal etc. This report focuses on the strategical decision making of a multi-agent soccer team for RCSS. Other studies have been done in the area of using reinforcement learning to train robocup robots as well as teams for the RCSS.

An interesting subproblem to the problem of strategical decision making during an RCSS match, where P. Stone et al have applied reinforcement learning, is keepaway [1]. This focuses on the problem of keeping the control of the ball within the team. P. Stone et al found that the training team’s performance had improved significantly after training.

In this report we implement and evaluate a multi-agent soccer team for RCSS. The team utilizes policy gradient reinforcement learning (PGRL) to train the strategie decisions of the agents. PGRL was chosen because it offers simplicities compared to other reinforcement learning methods. Traditional reinforcement learning methods have no convergence guarantees, while PGRL always converges to a local maximum [5]. Also, uncertainty in the state might degrade the performance of the PGRL policy, but optimization technique does not need to be changed. [3]

PGRL has been used for robocup soccer agents in multiple previous applications. An application relevant to ours is H. Igarashi et al’s study on the problem of
coordinating the kicker and receiver during free kicks [6]. They applied PGRL to this problem and found that if the kicker and receiver do not have the same heuristic, the receiver’s main focus will be to predict what the kicker will do. This leads to a master-servant relation where the kicker has all of the power and the receiver becomes a servant.

1.1 Problem statement

Can the performance of a multi-agent soccer team for RCSS be improved by applying PGRL to its strategical decision making process?
Background

2.1 RoboCup Soccer Simulator

The RoboCup Soccer Simulator is used to simulate a soccer game for robots on a two dimensional playing field. The simulator is turn based and works as a server which provides the players with the states of the game each turn. These states are updated every 100 ms and are a result of the actions that the players make.

The communication with the simulator is done via UDP/IP sockets; each packet sent to the simulator contains text that corresponds to a predefined set of commands. There are several communication protocols to choose from and they all differ in the message formation and content sent by the simulator. Each agent of the team connects to the simulator and initializes its simulated player. The goalkeeper is a special player and has to initialize himself as a goalkeeper for the server to recognize

![Figure 2.1. The different flags and lines that can be seen on the field.](image)
him as such.

Objects in the simulator reflect real objects often occurring in real robot soccer games, such as players or robots, a ball, and markers of the playing field.

Each agent is assigned one player in the simulator; with this player agents interact with the environment in the simulation. Through commands sent to the simulator an agent can control the designated player. The commands an agent can execute are described further down. Furthermore a player has stamina, recovery and effort. The stamina determines how much the player can run at maximum speed; when depleted, a player will be greatly slowed down. The recovery is how much the player regains stamina each turn. The effort determines how effective the use of stamina is.

Like a normal soccer game each team can have a goalkeeper. He has the same commands and abilities as a normal player except for the additional catch command. The catch command is unique for the goalkeeper and allows him to catch the ball. The area in which the ball becomes catchable is a rectangle in front of him with a length of two units and width of one unit. The ball in the game behaves similar to a real ball: it can be kicked in order to move it or caught by the goalkeeper. One important property of the ball is that it can move three times faster than a player. There are also numerous flags and lines on the field that can be seen by the visual sensor and are illustrated in more detail in Figure 2.1.

Some noise is added to the sensors to simulate real noise that may occur with real robots. Both the visual sensor and the movement of objects are subject to noise. The visual sensor of the players is sent as a message to the agent containing objects in the field of view and inside a small radius with less information. This field of view of a player is 90 degrees, 45 degrees on each side of the direction in which the head is pointed. The small radius is set to three length units. Objects inside it, but outside of the field of view, are shown with only distance to that object and what kind of object it is.

The players can hear and speak in the simulator. This is used to simulate a limited bandwidth communication between the agents. It is also used to get information about the game from the referee, such as state of play and ruling that the referee makes.

The last sensor used is the body sensor, a message about the body containing values of the player such as stamina, speed and head angle.

2.2 Controls

The players are controlled by commands sent to the simulator; some can be used only once per turn while others can be executed several times. The most relevant for this report are catch, dash, turn, move, and kick. Only one of these can be invoked each turn. The dash command makes the player move and uses stamina. The movement can be in any direction but it is most efficient to move into the direction of where the body is facing. The turn command makes the body of the player turn in any
direction. There are states of play that allow the player to teleport on the field to preserve stamina. The command for this is move. Using the kick command a player can be made to attempt to kick the ball, preferably when the ball is within kicking distance, which is 0.7 length units. A player cannot kick the ball if it is further away than the kicking distance.

The simulator has numerous settings that can be altered to change its behaviour such as message size limit, the visual sensor update interval and if a trainer is to be present.

There are several states of play that are similar to real soccer, the most common being kick-off, free kick and kick-in. The states of play are similar such that one of the teams has access to a radius around the ball and can kick it freely in any direction.

### 2.3 Rules

The rules are the same as a normal soccer game: the winner is the team with the most goals at the end of a match. Each match is 6000 turns and may be extended if there is a draw at the end of the original time.

The most common rules are also enforced by the referee such as free kick fault, offside, and backpasses. Free kick fault is a rule that prohibits a player from passing to himself during free kicks, kick-ins or kick-offs. Backpasses prohibit the goalie from catching the ball if it was kicked to him by a member of his team.

### 2.4 The trainer

It is possible to connect a trainer to the simulator. This trainer has some elevated rights, as opposed to a regular coach, and can control the play mode of the game. In addition he gets noise-free information about objects on the field and can communicate with the players on the field.
Method

The multi-agent soccer team was implemented in Java and run on RCSS version 15.0. Communication protocol version 13 was used.

In the server default settings; the agent would not receive a new state every turns. Since there is already uncertainty in the states due to the sensor model in RCSS, we chose to change the server settings. The message interval of the vision sensor was changed to 100ms so that the agent receives a new state each turn.

It is important to point out the difference between a player and an agent in our multi-agent soccer team. An agent controls a player. A player is an object within the simulator. Each agent will at every game state choose an action for the player. The actions are described below. The available actions in a given state depends on whether the player is in possession of the ball or not. There are also goalkeeper specific actions.

3.1 Formation

We opted for a 4:3:3 formation, i.e. one goalkeeper, four defenders, three midfielders, and three forwards. We have chosen not to go into depth about how the formation may affect the team’s performance. Therefore a simple formation was chosen. For more details see Figure 3.1

3.2 Actions available to player in possession of the ball

3.2.1 Dribble

Loosely kick the ball towards an area free of enemy players. Prefers to dribble towards the enemy goal.

3.2.2 Pass

Kick the ball towards a friendly player. Each possible pass, one for each friendly player, is regarded as an individual action. The power of the kick is a function of the distance to the player.
3.2.3 Shoot on goal

Kick the ball towards the goal with maximum power. The kick is aimed towards the largest free angle in the goal from the players perspective.

3.3 Actions for player not in possession of the ball

3.3.1 Go to ball

Run towards a stationary ball or intercept a moving one.

3.3.2 Hold formation

The player runs towards his position in the formation. The formation is moved with respect to the ball's position.

3.3.3 Cover opponent

Stay close to a specific enemy player. Each possible enemy player is regarded as an individual action.

3.3.4 Get open for pass

The player runs toward a location where it will be open for a pass.
3.4 Goalkeeper specific actions

3.4.1 Catch ball
The goalkeeper has the ability to catch the ball.

3.4.2 Cover goal
The goalkeeper runs towards a location between the ball and the goal and stays relatively close to the goal.

3.5 Decision making
As the focus of this report is learning for the strategical decision making of a multi-agent soccer team, the rest of this report will discuss how the player chooses between the actions above. Since the decision problem is highly complex, some generalization is needed. With PGRL the idea is to parameterize the policy [5].

3.5.1 Policy
The action is chosen by a policy. The policy is parameterized by a set of parameters that come from the current game state. From these parameters a heuristic value for each possible action is calculated. The action with the highest heuristic value is chosen.

We have chosen not to go into depth about how the heuristic values for each action are calculated and how we chose the parameters in question, since PGRL is guaranteed to converge [3], meaning that how the policy was designed is not important for the sake of this report. A team with a poorly designed policy will still be able to learn. However, a poorly designed policy might degrade the teams overall performance.

An interesting parametrization is discussed in Correlating Internal Parameters and External Performance by R. Nadella et al [7]: “When an agent wants to pass a ball to its teammate the decision whether or not to pass the ball is based on the following data: own passing skill level, the length of the pass, the skill levels and locations of the opponent players guarding the teammate to which the ball has to be passed.”. Our parameterization follows similar ideas, but we do not parameterize the skill level of any players.

3.5.2 PGRL
In order to be able to train our team, each parameter in the heuristic function for each action had a weight added. It is these weights our reinforcement learning algorithm will train. When we discuss exploration and learning with regard to the parameters it is not the parameters themselves that change. In reality the weights in our heuristic functions are changed to allow for the different parameters to affect
the teams behaviour in different ways. This will be further discussed in the policy gradient estimation part of this report.

**Reward system**

The goal for any reinforcement learning algorithm is to maximize the average amount of reward it gets in every time step. Therefore, it is important to choose a suitable reward system since this will greatly affect the team's behaviour.

An obvious approach is to reward the team for every match it won and punish it for every match it lost. Unfortunately this is not realistic in our case since every match takes at least ten minutes to play, we do not know how many iterations it will take for our reinforcement learning to converge, and the number of iterations will be further increased by the policy gradient estimation method we chose. Another downside to this approach is that team must be able to win some matches for the training to work.

A more short-term reward system would be to reward when the team scores a goal and punish when the enemy team scores a goal. The problem with this approach is that there is not a guarantee for goals in a soccer match. And if the team never manages to score a goal, the training will not work.

Even more short-term the reward system could reward things like the team being in possession of the ball, getting the ball into the enemy team's penalty box or being close to score a goal. Caution is needed when using short-term reward systems like these. Since the reinforcement learning algorithm will maximize the average amount of reward in every time step, the risk is that the team will become better at getting rewards, but not at winning matches.

The reward system we chose is a combination of rewarding goals and rewarding keeping the ball within the teams possession. The team is rewarded a small amount of points for every time step it remains in possession of the ball, and punished for every time step the enemy team remains in possession of the ball. The team is also rewarded a large amount of points for scoring a goal, and punished with a large amount of minus points when the enemy team scores a goal.

**Policy gradient estimation**

A common and simple policy gradient estimation method is the finite-difference method [3]. The idea is to explore parameters by increasing and decreasing them with a fixed step length. This can be further optimized by randomly generating a set of sets of parameters that vary with their respective step length. This is equivalent to saying that instead of exploring every combination of new parameters for a given policy, we randomly chose a few and explore those. The set that gave the most reward is chosen and the next iteration starts from there. In our case this means varying the weights in our heuristic functions. This will cause the agents to change their behaviour and allow for the training to take place.
3.6 Training

A trainer is used to train the agents. The trainer connects to the simulator and is granted elevated rights. It starts the matches and provides the agents with new parameters.

One iteration of the training takes place during one match. The match is divided into six episodes. For each episode a vector with new parameters is generated as described in the policy gradient estimation part of this report. Each of the episodes are evaluated based on the amount of reward earned during the episode, and the parameters that generated the most reward is saved and used as starting parameters for the next iteration.

The trainer manages this and stores the score and parameters from every match played.

3.7 Experiments

Our team was trained against four different enemy teams. Matches were played before and after training in order to measure our teams improvement. Results and parameters were saved for each match played during, before and after training. The teams our team trained against were: Non-learning team with same starting parameters: An exact copy of our team with the exact same starting parameters but no learning. Non-learning team with handpicked parameters: A copy of our team but with hand picked starting parameters. opuCl_2d[N]: A very good team. Competed in the 2D simulation league of RoboCup in 2008. TeamSkynet[I]: A very poorly playing team. Each team was played ten times before and after training. The training was done over 40 iterations.
Results

4.1 The team against its non-learning equivalent

The team trained against two instances of itself with different parameters.

4.1.1 Same starting parameters

The first opponent had the exact same starting parameters which stayed the same during all the training matches. Before the training both teams were equal and won five of the ten matches each. The ten first training matches the learning team won half of them, and after another ten matches it had improved to eight wins, one draw and one loss. Out of the ten following matches it won all of them. In the final set of ten matches it won nine and lost one. After training a set of successful parameters were chosen and set to static, this resulted in ten out of ten victories.

4.1.2 Handpicked parameters

The team with handpicked parameters had its starting parameters fixed to values we assumed would be performing fairly well. The handpicked parameters turned out to be good as the team won against the learning team ten out of ten matches before training. Of the 40 training matches the learning team won two and drew two. During the first matches the goal difference between the team with the handpicked parameters and the learning team was large. After training the learning team won one match and the team with handpicked parameters won nine out of ten. The difference in goals was a lot smaller and all of the matches were a lot more even.

4.2 The team against opuCl_2d

Before training, the team lost all of the ten matches against opuCl_2d. The learning team trained against the team opuCl_2d [9]. Of the 40 training matches played the learning team lost all of them without doing a single goal. The losses were consistent with the score fluctuating between 25 and 41. After training the learning team still lost ten out of ten matches against opusCl_2d. The difference in goals was not
notable. The behaviour of the learning team was notably more defensive after the training. The parameters from the training against opuCI_2d were used in a match against the team with the original parameters to observe the effects. This resulted in two defeats with 0:3 and 1:2, and one draw 2:2.

4.3 The team against TeamSkynet

Lastly the learning team trained against TeamSkynet \cite{4} and out of the 40 matches played it won all of them. The learning team won all of the ten matches both before and after the training. To observe the effects of training against TeamSkynet we used the parameters from the training and set the team to meet its non-learning equivalent with the original parameters. Though the matches were remarkably even, the trained team won both matches; the first with 4:3 and the second with 4:0.
Discussion

The learning teams performance improved against roughly equally skilled opponents. But against opponents that greatly outperformed the learning team or were greatly outperformed by the learning team the results were inconclusive. Below follows individual discussion of the learning team’s performance against the teams we trained and played against.

5.1 Training against its non-learning self with the same original parameters

The first ten matches started as expected, with the opposing team winning half of them, seeing as they are nearly identical except for small parameter changes that occur during PGRL training. After the first ten matches we can already see an improvement with eight wins, one draw and only one loss. This already shows that the learning improves the original parameter set. The following ten matches further strengthens that the PGRL improves the parameters. The last eleven matches the PGRL tried to go outside of what seemed to be a local maximum and lost two matches.

5.2 Training against non-learning self with handpicked parameters.

Unfortunately the learning team converged to a local maxima that was not good enough to beat the team with handpicked parameters more often than it lost. The team with the handpicked parameters played in a very structured way and the learning team struggled to get the ball far into the opponent’s half of the field. The learning team did manage to defend better, losing with less difference in goals after the training, and winning rarely. But since the learning team did not manage to score many goals in any match, the correlation between scoring goals and getting reward must have been much weaker than the correlation between defending well and getting more reward. Since we rewarded goals and punished enemy goals this result is expected.
5.3 Training against opuCI_2d

The training matches against opuCI_2d were one sided. The implemented team was inferior in every aspect. It passed, dribbled and intercepted worse than the opponent team. The action methods were far too inadequate for the parameters to make a difference. This is shown by the consistent loss with approximately the same score each time.

Using the parameters from the training to meet the the team with the original parameters resulted in defeat. The training against opuCI_2d resulted in the team focusing more on passing and holding position, meaning that there was no forward play, with the game mostly staying in the middle of the field with occasional breakout by a single opponent player that dribbled and shot on goal.

5.4 Training against TeamSkynet

The matches against TeamSkynet were mostly controlled by the learning team as their players mostly stood still. Their goalkeeper stood still through the whole match, he only responded if the ball was shot directly at him by stopping it. The backs stood still until the ball reached a small radius and would return to their original position when the ball exited the radius. The forwards did most of the running but did not succeed in intercepting the ball. Some players froze after some time and the team did not handle game states such as kick-ins and kick-offs in mid game. All of this makes the results against TeamSkynet uninteresting and useless.

When we had the team with these parameters play against an equivalent team but with the original parameters, it was apparent that the playing characteristics had been affected in such a way that the goalkeeper mostly stood still, and the players were more prone to chasing the ball, attempting to shoot for goal from a further distance than before. This resulted in an effective way of beating the original team.

5.5 General

Although it was difficult to measure the learning team’s performance when training against opuCI_2d, the learning teams behavior did change. It became more passive and defensive. It would seem like a more defensive strategy gave more reward. Keeping the ball as long as possible by passing and holding position might have resulted in fewer enemy goals or more possession of the ball. The results against the poorly playing TeamSkynet were also inconclusive. Maybe the initial parameters were close to a local maxima. Possibly TeamSkynet played so poorly that any change in behavior in the learning team still resulted in a relatively equal amount of reward. However, the change in behaviour shows that the agents are trying to maximize their reward. Unfortunately they do not have the basic prerequisites needed to become a very good team.
5.5.1 Improvements

It might have been more efficient to take an already playing implementation and parametrize its decision making and use PGRL on it. The methods implemented can be improved significantly in order to make the teams' performance better. Although it would not have changed our result, had we had access to a very good team's actions, our team's overall performance would have improved a lot.

The team's ability to intercept a moving ball or intercept a stationary one inefficient in such a way that the players makes between one and four extra turns, but mostly only one. This results in a slowdown each time the player turns instead of dashing because of the speed decay in the simulator.

The team passes very accurately, the shooting on goal is sufficiently effective and interception of the ball is good. Its not enough to really make it a great team. There is room for improvements, such as holding formation and positioning of the player to receive a pass.

Our team also seemed to struggle with progressing deeper into the opponent’s half of the field. There are different ways to improve this. A simple action could have been added that moved the player further into the enemies half of the field, a parameter that moved the formation more aggressively forward could have been added or the hold position action or get open action could have been changed so that the players assumed more aggressive positions.

Our implementation could also benefit from a more sophisticated evaluation for passing and holding position. Some parameters that represent how likely it is that the ball will be intercepted by an opponent and if the player intended for the pass has an advantageous position could be added. Evaluating how, when and where to hold your position is very difficult and could be improved greatly compared to our implementation.

The formation could probably be improved on as well. Different formations could be tried, distances between players and the formations relative placement on the field could be varied.

5.6 Future work

5.6.1 Different reward systems

An interesting future study would be to evaluate the difference between different reward systems. Are certain reward systems better against certain other reward systems? Will a more short term reward system outperform a more long term one due to its flexibility or will a more long term reward system eventually find a better local maximum? An example would be to compare a team with a reward system based on score and a team with a reward system based on possession of the ball.

With heavy parallelization and better hardware it might be possible to run enough simulations in a manageable time to evaluate a reward system based on very long term performance such as tournament placements. This could help to
eliminate some of the fluctuation caused by randomness in the simulator.

5.6.2 Advanced handling of states of play

It would also be interesting to incorporate separate learning for free kicks, kick-offs and kick-ins. These states differ from normal states during a match because the ball is not in play during them. If your team is guaranteed to get to the ball first and get a free kick with enemies at a certain distance, a more aggressive decision making might be more rewarding. However, when the enemy has a free kick you might want to assume a more defensive strategy that focuses on covering enemy players or the goal etc.

5.6.3 Player roles, formation and more.

Another interesting change would be to classify the agents as different roles and train them individually. This would allow us to train forward players to be more aggressive while backward players would still be able to play relatively defensively.

Player roles could be combined with trying different formations. Maybe including more forward players is better against certain teams etc. Maybe the team would benefit from changing the formations during play depending on different parameters.

A development of these ideas could be to let the number of certain roles and the formation change depending on the current score of the match and the time left. An example would be to make some of the players get the forward player parameters and a more aggressive formation during free kicks and kick-ins close to the enemy goal. All of these things would be interesting to evaluate.
Conclusion

PGRL applied to the strategical decision making of a multi-agent soccer team in RCSS did improve the performance of the team. All of the trained teams changed their behaviour in some way. The team that played a much better team assumed a more defensive strategy, the team that played a poorly playing team assumed a more aggressive strategy. Generally the team tended to pass more after training. When the gap in performance between the learning team and the opponent team was too big, the results were inconclusive. A downside, and upside, to PGRL is that it converges towards a local maxima. Even if the learning team has the potential to outperform the team it is trained against, it might find a local maxima that is not good enough for the learning team to reach its full potential. But since we know that PGRL does converge towards a local maxima, the performance will always improve if we are not already in a local maxima. The most relevant result is that PGRL can significantly improve strategic decision making for a team when trained against a roughly equally skilled team.
Bibliography


downloads/48107/opuci_2d-robocup2010.tar.gz/ Last accessed 2014-03-11