Learning for RoboCup Soccer
Policy Gradient Reinforcement Learning in multi-agent systems

Christian Lidström    Hannes Leskelä
clid@kth.se    hleskela@kth.se
860425-4972    910807-0997

2014-04-13

KTH - Royal Institute of Technology,
CSC - School of Computer Science and Communication
DD143X - Degree Project in Computer Science, First Cycle
Supervisor: Christian Smith
Examiner: Örjan Ekeberg
Abstract

RoboCup Soccer is a long-running yearly world wide robotics competition, in which teams of autonomous robot agents play soccer against each other. This report focuses on the 2D simulator variant, where no actual robots are needed and the agents instead communicate with a server which keeps track of the game state.

RoboCup Soccer 2D simulation has become a major topic of research for artificial intelligence, cooperative behaviour in multi-agent systems, and the learning thereof. Some form of machine learning is mandatory if you want to compete at the highest level, as the problem is too complex for manual configuration of a teams decision making.

This report finds that PGRL is a common method for machine learning in RoboCup teams, it is utilized in some of the best teams in RoboCup.

The report also finds that PGRL is an effective form of machine learning in terms of learning speed, but there are many factors which affects this. Most often a compromise have to made between speed of learning and precision.
Referat

RoboCup Soccer är en årlig världsomspännande robotiktävling, i vilken lag av autonoma robotagenter spelar fotboll mot varandra. Denna rapport fokuserar på 2D-simulatorn, vilken är en variant där inga riktiga robotar behövs, utan där spelarklienterna istället kommuniserar med en server vilken håller reda på speltillståndet.

RoboCup Soccer 2D simulation har blivit ett stort ämne för forskning inom artificiell intelligens, samarbete och beteende i multi-agent-system, och lernetet däremot. Någon form av maskininlärning är ett krav om man vill tävla på den högsta nivån, då problemet är för komplext för att beslutsfattandet ska kunna programmeras manuellt.

Denna rapport finner att PGRL är en vanlig metod för maskininlärning i RoboCup-lag, den används inom några av de bästa lagen i RoboCup. Rapporten finner också att PGRL är en effektiv form av maskininlärning när det gäller inlärningshastighet, men att det finns många faktorer som kan påverka detta. Oftast måste en avvägning ske mellan inlärningshastighet och precision.
# Table of Contents

1 Introduction ................................. 5
   1.1 Problem statement .......................... 5
   1.2 Approach .................................. 5

2 Background .................................. 6
   2.1 RoboCup .................................. 6
      2.1.1 RoboCup Soccer 2D simulation .......... 6
      2.1.2 Server .................................. 7
      2.1.3 Clients .................................. 7
      2.1.4 Monitor ................................. 7
   2.2 Machine learning ........................... 7
      2.2.1 Reinforcement learning .................. 8
      2.2.2 PGRL .................................. 8
      2.2.3 Simple PGRL algorithm .................. 8
   2.3 RoboCup Teams ............................. 8
      2.3.1 WrightEagle ............................ 8
      2.3.2 Team Skynet ............................ 9
      2.3.3 UT Austin Villa ......................... 9

3 Methods .................................... 10
   3.1 Implementation ............................ 10
      3.1.1 Implementation details .................. 10
      3.1.2 Parameterization ......................... 11
      3.1.3 PGRL .................................. 11
      3.1.4 Coach mode ............................. 12
   3.2 Review of other teams and research ........ 12

4 Results .................................... 13
   4.1 PGRL in RoboCup teams ..................... 14

5 Discussion ................................. 15
   5.1 Optimal values ............................ 15
   5.2 PGRL ...................................... 16

6 Conclusion .................................. 17

Bibliography .................................. 18
1 Introduction

Within our programme at KTH, the first cycle degree in computer science is the culmination of our first three years of studies, where we utilize our acquired skills and knowledge to independently conduct research on a specific field and problem. Furthermore, we are to critically and independently evaluate our result as well as the result of others. [13]

The general subject of this paper is to study the use of policy gradient reinforcement learning (PGRL) in multi-agent systems, with a case study of PGRL applied to RoboCup Soccer. RoboCup Soccer is an annual soccer tournament for multi-agent systems, where teams of 11 agents face off against each other. Agents can only communicate via certain channels and what is apprehended via the agents input systems, making some sort of decision based AI crucial to a well-functioning team.

We will look at the effectiveness of policy gradient reinforcement learning in RoboCup Soccer to gain a better understanding of its advantages and disadvantages in multi-agent system. PGRL is less complex relative to many other forms of machine learning, and commonly used in these sorts of multi-agent systems.

This is an interesting subject in general, seeing as more and more systems are becoming autonomous. Things from cars to assembly line robotics or web crawlers all utilize some form of AI to perform their operations.

1.1 Problem statement

The problem to be investigated is if PGRL is a suitable and effective technique for machine learning in multi-agent systems, specifically in RoboCup Soccer, with respect to the length of the learning period, which we define as the number of iterations it takes for the algorithm to find an optimal set of values.

1.2 Approach

In this project we will be using the RoboCup Soccer Simulator software, which is a simplified 2D simulator of RoboCup Soccer. This lets us put emphasis on AI and team behavior in multi-agent systems rather than the technical details of robotics.

We will make a simple implementation of PGRL in a RoboCup Soccer team, through parameterizing a small subset of the decision making process, in order to analyze its effectiveness.

Because of the time constraint this implementation will not be very complex, and the results of it will be combined with studying previous literature on the subject, and researching the most successful RoboCup Soccer teams and their approach to machine learning.

In order to evaluate PGRL in the RoboCup Soccer environment we will focus on examining its speed of learning.
2 Background

2.1 RoboCup

RoboCup is a long running initiative whose origins dates back as early as 1992. It was in an essay that professor Alan Mackworth discussed the idea of seeing robots, and robots playing soccer, albeit in a primitive fashion. [1]

Concurrent with his essay a Japanese research team proposed and defined the rules and goals with a robotic soccer league, which became a reality in 1993. This was called Robot J-league after the Japanese soccer league, but after a series of global requests the name was changed to “the Robot World Cup Initiative”, abbreviated as RoboCup. [4]

However, it was not until 1997 that the official league started with both a simulated 2d league and a real RoboCup soccer league that layed the foundation for the current RoboCup leagues. A large number of papers surged from this event, spanning from reinforcement learning in multiagent systems to optimization of robot movements. [18]

These motivated the continued efforts to develop the tournament, whose ultimate goal is to “By the middle of the 21st century, a team of fully autonomous humanoid robot soccer players shall win a soccer game, complying with the official rules of FIFA, against the winner of the most recent World Cup.” [5]

However, the less extravagant goal is to, in general, increase the interest for AI and robotics research via a simple yet formidable challenge.

2.1.1 RoboCup Soccer 2D simulation

The 2D soccer simulation league was the first event of the RoboCup soccer simulation league. The game is run by a server that is based on the works of the Japanese scientist Itsuki Noda, who conducted research on multi-agent systems during the 90's. The league is an annual event taking place in different cities across the globe, most recently in Eindhoven, Netherlands where the highly successful team WrightEagle won their 4th gold. [8]

2.1.2 Server

The server is what provides the soccer simulation. It keeps track of the game state and updates it according to players actions, its physics and the rules of the game. The simulation updates every tenth of a second, during which players recieves some information about the game state, makes decisions on what to do, and sends the server these actions. The server then handles it according to the physics and rules of the simulation. For example, a player might request to sprint but not have enough stamina, and will therefore only run slowly, or the ball might have been kicked into one of the goal, and the game will award 1 point to the scoring team and call for a kick-off. A game of RoboCup Soccer consists of 6000 of these game ticks, which make for 10 minute long matches. [9]

There is however also a set of fair play rules, which the server cannot control, and thus a human referee is also needed for real matches of RoboCup Soccer. [3]
2.1.3 Clients

There are three different types of clients able to connect to the RoboCup Soccer Simulation server: the 11 player agents of each team, one coach per team, and one special kind of coach, that is not allowed in real matches, called the trainer.

The player agents receive 3 types of information from the server, representing body, aural and visual sensors. These are the only ways each player agent can receive information about what is happening in the game. The body sensor provides information such as the player’s current position, speed and stamina.[3] The aural sensor sensor receives what the player can hear, like messages from the referee or other players. The visual sensor gets information about what the player can see, such as the position of the ball and other players. This information is somewhat distorted depending on the distance to the things players can perceive. For example, a player might know the exact position and identity of a player standing right next to him, but the position might off by several meters to someone standing half a field away, and it might not even be possible to make out what team that player belongs to. The players can then act on this information by sending the server a request of what it wants to do, like kicking, running or talking.[3]

The regular coach agent has the advantage of receiving the exact state of everything on the field, which allows the coach to make better decision about what actions to take. However, it has the disadvantage of more restricted communication with the players, in order to prevent centralization of the entire teams decision making.[12]

The trainer, which can only be used in training sessions for the purpose of development, is much more powerful than the ordinary coach. It has unrestricted communication with the players and can freely change the game state. This is the main tool when developing a team using machine learning, as it allows for setting up automated training sessions to run simulations that the agents can learn from.[12]

2.1.4 Monitor

Also included in the RoboCup Soccer 2D Simulation package is a monitor program which is used to visualize the simulation for human spectators. It draws the virtual football field, the ball and all the players and updates every tick according to their movement, and displays information such as the current score.[3]

Also, it offers a simple interface for manually changing the game state, allowing human spectators to call for kick-offs, free kicks, red cards and so on.[3]

2.2 Machine learning

Machine learning is a concept often used in artificial intelligence systems, and aims to give computer programs (in this case intelligent agents) the ability to learn without being explicitly programmed.

It usually works by feeding the program a large data set from which it can generalize methods on how to best approach new tasks in the future.[19]
2.2.1 Reinforcement learning

Reinforcement learning is a subset of machine learning which focuses on how intelligent agents can best make decisions on what actions to take in a certain environment in order to maximize some predefined notion of reward. [14]

A major difference in reinforcement learning compared to most standard machine learning techniques is that there is no correct output given for the input in the learning phase, and that there is no inherently optimal action in any given subtask. [10]

This makes reinforcement learning more of an explorative technique, and fits the naturally undecidable nature of agent versus agent competitions like RoboCup Soccer, where no optimal play can clearly be defined.

2.2.2 PGRL

Policy Gradient Reinforcement Learning is a form of reinforcement learning which utilizes policy gradient methods. This means they rely on maximizing a cumulative reward by optimizing the values of previously parameterized policies, using gradient descent (or ascent). [6]

In other words, given a parameterized policy, the agent continuously try to run the policy with new values that it extrapolates from previous runs as having a positive effect on the end-result. [6]

PGRL is guaranteed to find a local optimum, but there is no way to know if this is also a global optimum, which is one of the biggest disadvantages. [1] If this is a major concern, one way to combat it is to make jumps to other values in the parameter space and continue running the algorithm from that point. This increases the likelihood of finding many different local optima that can then be evaluated.

2.2.3 Simple PGRL algorithm

This is a description in simple terms of how a basic implementation of PGRL could work.

Start by parameterizing the decision making process, arbitrarily choose a set of starting values, and assign a delta value for each parameter to change it with after each evaluation of the policy. Then decide on a reward function that will evaluate the results of the decisions made using the current policy parameters. Run the simulation with the starting parameter values, and calculate the reward.

Now, iterate over the parameters one by one, change them according to your delta value, run the simulation again and calculate the new reward. If the parameter had a positive effect on the reward, keep changing it in that direction, if it had a negative effect change it in the opposite direction, until an optimal reward value has been achieved.

2.3 RoboCup Teams

Here follows a short description of RoboCup teams relevant to this report.

2.3.1 WrightEagle

WrightEagle is made by the Multi-agent Systems Lab, a group of researchers at the University of Technology of China [8].
It is one of the most successful teams in RoboCup Soccer 2D simulation competitions, which naturally means that their methods of having their agents learn must be among the best. From the research papers they have released on machine learning in RoboCup Soccer [8], we can see that a common theme is to utilize PGRL. For example, in one paper PGRL was applied to maximize the success rate of free kicks [6]. This indicates that machine learning in RoboCup Soccer, a multi-agent system with complicated rules, is best applied to a small subset of the game. If the scope of the learning algorithm is too broad it will be hard to get any relevant results since the parameter space will too vast.

Furthermore, in team WrightEagle PGRL has been used to improve decision-making in most of these kind of subsets of the game. Their use of it indicates that PGRL is one of the best methods for making your team make good decisions.

There is a fully open-sourced version of WrightEagle called WrightEagle Base, which is the basic implementation of WrightEagle without any machine learning techniques implemented, meant to be used for educational purposes.

2.3.2 Team Skynet

Team Skynet is an open-source RoboCup team written in Java and made by two students from the University of Kansas [15]. As such, it performs at a much lower level than that of WrightEagle.

Team Skynet also had several minor bugs. Even after fixing the bugs, Team Skynets performance would still be subpar compared to that of WrightEagle Base, and would need more than having machine learning implemented in order to be truly competitive.

2.3.3 UT Austin Villa

UT Austin Villa is a prominent RoboCup 2D and 3D simulation team, as well as a high ranking team in the standard platform league. [2] The team is created by researchers at the University of Texas at Austin. They are the current 3D simulation league champions [16], and have released several reports of their PGRL research. [17]

Although we have not used the team in our simulations it is featured in our literary research. For instance, they created the subtask keepaway in RoboCup Soccer 2D to have a test bed for reinforcement learning, where their policies are credited as better than the benchmark policies they were compared to. [14]
3 Methods

The methods we used to evaluate the effectiveness of PGRL in multiagent systems can be divided into two distinct parts. The first one is to implement our own team and apply machine learning in the form of PGRL to it.

The second one is to research the use of machine learning in existing teams, which methods they use and what their results have been. This also includes looking at previously done research on machine learning in RoboCup, not necessarily tied to a specific team.

What we are trying to accomplish, as stated in the problem statement, is to investigate if PGRL is an effective machine learning method in multi-agent systems with respect to learning speed. With the above mentioned approaches we believe that we can confirm the effectiveness both with our own implementation and by showcasing the success of other teams. It is not necessarily so that PGRL is the winning factor in robocup soccer, since the choice of parameters in the different game situations affects the decision making more than the learning algorithm. But we believe that if enough high ranking teams utilizes PGRL there must be a reason behind it.

3.1 Implementation

To simplify our task and focus on a more interesting aspect of machine learning we decided to base our implementation on an existing team with open-sourced code instead of writing our own from scratch. We did this in order to allow us to focus primarily on the machine learning aspect. We did research to find the source code of several teams, and decided which team to use based on the code it was written in. The team we eventually chose to implement machine learning, Team Skynet, was chosen because of our familiarity with the source language and that the source code was relatively easy to understand. Preferably, the WrightEagle Base team would have been used, but the code was too extensive to grasp in the timespan we had.

We also spent some time fixing the previously mentioned bugs in Team Skynet before utilizing it in our experiments.

Furthermore, we needed to implement our own coach to actually perform some kind of learning. More on this in the Coach mode section.

3.1.1 Implementation details

We believe that matching our revised Team Skynet against either the old version or WrightEagleBase and base our success rate on a win or loss or the amount of goals made would be a poor decision. This is because game outcomes depends on small random variances in the game logic. When plotting the same two teams against each other, scores can vary greatly and even the winning team differentiate. Statistically the outcome would strive towards both a default winning team and goal difference, but we believe that the changes we make in each PGRL iteration wouldn't affect the result enough to make it our basis for determining improvement.

Therefore we will focus on a specific subset of the game, much like has been done in other teams, e.g. WrightEagle. The subset we have chosen to focus on is kicks just outside the opposing teams' penalty area, and their success rate
when it comes to scoring goal versus a goalkeeper. We have also chosen to use the goalkeeper of WrightEagle Base in our experiments, in order to teach our player to score on a more competent opponent.

The subset of RoboCup Soccer that we chose to apply PGRL to is very specific in that it only allows for the player to improve its decision making when it comes to trying to score a goal versus a single opponent, the goalkeeper. It also means the player will only learn how to shoot better. In a real game, where there are more factors involved, it might be a better decision to dribble or pass. Thus our learning efforts will only apply when the best decision actually is to shoot from your current position. Thus, we can't tell how much this specific improvement actually affects the overall performance of the team.

This choice of specific subset of the game also have several advantages. It won't let the overall subpar performance of Team Skynet affect the outcome too much, if at all. It also means the parameter space will be relatively small, which should let us make improvements quicker. This is important because of the relatively small scope of this project.

3.1.2 Parameterization

Before we can apply our PGRL to our team we had to find a way to parameterize the behaviour of the player in this specific situation. While our learning situation is extremely limited, we still want to make the player learn how to best take shots from different angles and slightly different distances. This means there is still room for generalizing our parameters.

The parameter space we decided to use consisted of kick power, minimum shooting angle from the goal posts, minimum shooting angle from the goalkeeper, and ball placement in between these two angles. The kick power is a numerical value between -30 and 100, where 100 is the hardest a player can kicking. A negative value implies kicking backwards. The two angular parameters are the smallest offset at which the player is allowed to aim from the goal post and goalkeeper respectively. The ball placement parameter is a value between 0 and 1 which specifies where the player will aim between the two angular offsets. A value of zero implies shoot directly at the angle offset from the goalkeeper, and vice versa. These parameters will naturally also be dependent on the positions of the player, the goalkeeper, and the goal, allowing the player to learn a generalized set of parameters.

3.1.3 PGRL

In our PGRL implementation, we decided on a reward function that only give points for successfully scored goals, and subtract a slightly lower amount of points for completely missing the goal with a shot. This is because in a real game a kick towards the goal which is blocked by the goalkeeper is still better than completely missing the goal, since it can be followed up by other players. The exact values used were 3 points per scored goals and -1 for missing the goal.

We decided on using starting values that we thought would be close to a local maximum. So we started with the maximum value for kick power, and with direction offset parameters resulting in a shot in the middle of the goalkeeper and a goal post. If the kick is too close to the goal post, the randomness of the players vision and the shots direction can cause it to miss the goal. The
maximum value for kick power was chosen because it should be close to the optimum value. A slow ball is easier to catch, and the only thing that might affect a high kick power negatively is that a shot might be less precise and therefore miss the goal more often.

We also used relatively big delta values, because very small differences in the parameters did not lead to any significant results, since the random factors (like imprecise kick direction) would be too great compared to our delta values. This would result in the algorithm changing the values around randomly. The delta values we used was 10 for kick power, 5 degrees for both the offset angles, and 0.1 for the target to aim at between the offset angles.

For each parameter setup we ran 100 simulations, where one simulation is one shot, on three different locations outside the penalty area. This was to ensure that the result of our simulations had some practical usage in real game situations, where not every shot is taken at an advantageous angle such as straight in front of the goal. The player is fixed to these positions to ensure that the kicking is in focus and not dribbling combined with kicking.

3.1.4 Coach mode

To monitor and train our team we decided to utilize the coach mode, a.k.a. training mode, included in the RoboCup Soccer Simulation package. The offline coach has all the real world location data and can inform players of objects position relative the player without any time restrictions that the online coach has. Our coach utilizes the fundamental parsing capabilities that our players implement to send and receive messages from the server. The messages sent are either a look, a move or a change_playmode command to keep a certain training routine active. The players and ball are reset if they leave the training area, and we utilize sleep functions in conjunction with different playmodes to keep the players stationary pre exercise.

3.2 Review of other teams and research

To evaluate the effectiveness PGRL we can not only look at the results of our own experiments, since the scope of this project is too limited to confidently assess its performance. Because of this, it is reasonable to study what techniques top RoboCup teams use to improve upon their abilities, and general research done at a higher level in the area.

In doing this evaluation, we have looked at usage of PGRL in top-level teams as a machine learning technique, and if speed of learning has been taken into consideration.
4 Results

These are the results of running our PGRL algorithm for the previously defined parameter space.

As mentioned in the method-section, our first runs of the PGRL algorithm ended up never converging to an optimal value. This was because our values for parameter changes were too small, which resulted in the random factors inherent to the game having more effect than the change in parameter values. This lead to us increasing the delta-values in order for the algorithm to converge.

The data presented here is from the final run of the PGRL algorithm. The parameter values were updated between each iteration, and evaluated according to the reward function after each iteration. The PGRL algorithm was run until it had found an optimal set of parameter values, and as previously stated each iteration consisted of 300 simulations from 3 different positions.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>KP</th>
<th>GPO</th>
<th>GKO</th>
<th>TBO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Results</th>
<th>SOG</th>
<th>SP</th>
<th>MS</th>
<th>TS</th>
<th>Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position 1</td>
<td>100</td>
<td>90</td>
<td>0</td>
<td>100</td>
<td>270</td>
</tr>
<tr>
<td>Position 2</td>
<td>88</td>
<td>77</td>
<td>12</td>
<td>100</td>
<td>219</td>
</tr>
<tr>
<td>Position 3</td>
<td>74</td>
<td>51</td>
<td>26</td>
<td>100</td>
<td>127</td>
</tr>
<tr>
<td>Combined</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>616</td>
</tr>
</tbody>
</table>

Table 1: Data from the first iteration

<table>
<thead>
<tr>
<th>Parameters</th>
<th>KP</th>
<th>GPO</th>
<th>GKO</th>
<th>TBO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Results</th>
<th>SOG</th>
<th>SP</th>
<th>MS</th>
<th>TS</th>
<th>Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position 1</td>
<td>100</td>
<td>83</td>
<td>0</td>
<td>100</td>
<td>249</td>
</tr>
<tr>
<td>Position 2</td>
<td>100</td>
<td>74</td>
<td>0</td>
<td>100</td>
<td>222</td>
</tr>
<tr>
<td>Position 3</td>
<td>97</td>
<td>76</td>
<td>3</td>
<td>100</td>
<td>225</td>
</tr>
<tr>
<td>Combined</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>696</td>
</tr>
</tbody>
</table>

Table 2: Data from the last iteration

The first part of the data describes the input values, with columns Kick Power, Goal Post Offset, GoalKeeper Offset, Target Between Offset.

The second part of the data describes the outcome and reward of the simulations in three different positions, with columns Shots On Goal, Scored Points, Missed Shots, Total Shots.

The reward was then combined for the three positions to get a sum, which is what was being maximized by the PGRL algorithm.
4.1 PGRL in RoboCup teams

Our literary study did not yield any results regarding the use of PGRL for faster learning. Although speed as a variable is not mentioned, the top team in both the 2D and 3D simulations league utilize PGRL, as well as the leading team in the standard platform league. [16] [8]
5 Discussion

This is a discussion of our results and the methods which produced them.

5.1 Optimal values

The results of running our PGRL algorithm were both expected in some ways, and surprising in others. What was expected was that our starting values were not very far off from a local maximum. We had chosen the maximum value for kick power because it should be close to the optimal value, and it turned out to actually be the optimal value in the simulated situations.

We also started with values that would place the shot halfway between the goalie and a goal post. We had guessed that this might be close to the optimal point, but also thought that the actual best point to aim at was probably further out towards a goal post, to decrease the chance of the goalkeeper catching it. The result was surprising, in that the best point to aim at in reality was closer to the goalkeeper than a goal post. This could be explained by the fact the when aiming near a goal post the random noise factors inherent to the game would cause the shot to miss the goal completely. Examples of random factors in this case would be that the players sensory information is not perfect, which leads to a slightly distorted perception of where the goal really is, and also that shots do not go in perfect straight line.

The parameters controlling the maximum offset of where to aim in comparison to the goalkeeper and goal posts turned out not to have any real significance. This was somewhat expected, since the last parameter controlling where in the interval specified by the two offset parameters the shot would be aimed at has a larger impact on the outcome.

We had some surprising results on previous runs, which can be attributed to random variances in the game. A few of our early runs had much smaller delta values, which led to the algorithm sometimes not getting anywhere, and as such we had to increase the delta values. This has the unfortunate effect that the results might be a little less precise. This could probably have been somewhat prevented by running the simulation many more times each iteration.

Our reward function (which awards 3 points per scored goal, 0 when the goalkeeper saves the ball, and -1 when the player misses the goal completely) was chosen because scoring a goal is the primary goal, but getting a shot off towards the goal is still better than missing it, especially in a real game where this will have effects on the game moments after the shot. This will affect which optimal values will be found, since it might actually favour less total scored goals, if that also leads to the number of missed shots is reduced by an amount more than a factor of three than the change in scored goals.

Another thing to take into consideration is that we cumulatively reward the player for simulations from three different shooting positions around the penalty area. This means that the optimal values we have found is based on an average of these three positions, and might not be the optimal values for any one position.
5.2 PGRL

Our PGRL-algorithm needed only 21 iterations to converge in an optimum in our final run, which is a surprisingly small amount. This is of course the result of the relatively large delta values, which as discussed earlier are necessary unless we ran the algorithms with a number of simulations many magnitudes larger. Another way of ensuring a precise result, which also wouldn’t work in our specific simulation, would be to decrease the delta values the closer you are to an optimum, and increase them when far from it.

The small set of parameters also greatly affects the running speed of the algorithm, as each additional parameter would exponentially increase the time it takes to search the parameter space for optimal values. We chose the small set of parameters because of the limited scope of this project, and it fit the chosen subset well.

PGRL will only give us local optimum, [7] but because of the limited subset of the game we are examining, this can be confidently said to be the global optimum. Because of our choice of parameterization there might of course be other sets of parameter values that are just as good, but the end result (i.e. where the ball is aimed) will still be the same. Had we tried to apply the same algorithm to a larger subset of the game, we would have had to be careful to not find a local optimum that is very far from global optimum.

This report was meant to find out whether PGRL was an efficient method of machine learning in multi-agent systems in terms of the length of the learning period. Even though our parameter space was very limited and our delta values relatively large, we think our results show how great PGRL is in terms of learning speed.

Furthermore, there are lots of ways to implement PGRL, and our implementation is certainly one of the less complex. While this is not bad in its own, it might mean we got an unusually fast algorithm, and that more complex variants on PGRL will be slower, and hopefully more precise.

Since PGRL is used in some of the best RoboCup teams, e.g. WrightEagle [8] and UT Austin Villa [16], and our implementation of PGRL not only found an optimal value quickly, but also showed that it can be adjusted to any desired learning speed, we feel that a reasonable conclusion is that PGRL is, aside from its less complex nature, an efficient solution for machine learning in terms of learning speed.

Lastly, we would like to note that our experiment has only improved the players ability to score goals from just outside the penalty area, which might not mean that the teams ability to score goals in a real game has improved at all. First of all the team we based our development on would probably still have great problems even getting close to the penalty area. Second, actually taking the shot might not always be the best decision. Even though our player can score goals consistently during our practice setup, there might be situations where passing the ball is the better option in a real game.
We have found through research and experiments that using PGRL for machine learning in multi-agent systems can be an efficient solution in terms of agents learning speed.

However, there are many factors which can impact this rate of learning. If speed of learning is of importance care has to be taken to choose a good parameterization. The parameter space can not be overly large, or it would greatly impact the speed at which an optimum can be found. Another important factor is the size of delta values used to travel along the gradient. The more precise results you want, the slower the algorithm will be.

Another important consideration should be what starting values to choose. This should also affect the speed of learning, although not by as much as the previously mentioned factors. The more important consideration to have when choosing starting values is what optimum you will find, since PGRL only finds local optimums.
Bibliography


[13] Royal Institute of Technology. Dd143x degree project in computer science, first cycle 15.0 credits [internet; course webpage], 2014 [Retrieved April 10, 2014]. http://www.kth.se/student/kurser/kurs/DD143X/?l=en


