Learning Coaching Advice to improve playing skills in RoboCup

Eva Bou, Enric Plaza and Juan A. Rodríguez-Aguilar
IIIA - Artificial Intelligence Research Institute,
CSIC - Spanish Council for Scientific Research,
Campus UAB, 08193 Bellaterra, Catalonia (Spain).
{ebm,enric,jar}@iiia.csic.es

ABSTRACT

Coaching is a way to help an agent or a group of agents to learn and/or to improve his/their performance, but learning to coach is a tough challenge. In this paper we try to make headway in this matter by presenting a learning method that uses decision trees to learn pass advices from observations of players' actions in the simulated RoboCup soccer environment. We propose and evaluate different learning techniques to build decision trees to from which passing advices can be generated. Finally, we empirically demonstrate that exploiting learnt advices can significantly improve the number of successful passes between teammates.

1. INTRODUCTION

RoboCup [1] is a distributed, multiagent, cooperative, competitive domain that is used for research in multiagent systems. The simulated robotic soccer domain enables two teams of eleven simulated autonomous robotic players to play soccer. Futhermore, it allows to include a coach agent per team acting as an advice-giving agent. A coach agent receives global and noise-free information about the objects on the field at all times during the game. He is an agent that can provide advice to playing agents in order to improve their performance. In RoboCup an advice is a rule of type "if condition do/dont action" in a language called Clang [1]. The players may take into account the advices from their coach for their decision making during a game.

It is a challenge to create a coach that improves significantly a team's performance by providing advices.We believe that a coach can improve a team's performance if he adapts his advice to the team's behavior and skills. This work focuses on learning advices for coach agents to help increase the number of successful passes between teamates. This is achieved by advicing players whether to pass the ball or not depending on the situation of the game at play. At this aim we depart from a machine learning technique to learn advices from observations of passes between teammates. We use decision trees to learn advices because they are easy to transform into Clang advices and also because the structure of a decision tree is easily understandable by humans.

We experiment with different techniques for building a decision tree and with different heuristic measures to perform the classification between good and bad passes. Futhermore, we perform an empirical evaluation of each technique and measure. Thus, we evaluate the effects of the learning advices on the players in two experiments. Finally, the results of evaluating the pass performance of the players with the advices obtained from combining each technique with each measure show that the advices significantly increase the number of successful passes between teammates: from 9% up to 19% depending on the combination of technique with measure used for building decision trees.

The paper is organized as follows. In section 2 we summarize the related work. In section 3 we briefly introduce the simulated robotic soccer platform focusing on coaching. Then, in section 4 we present the scenarios that we use for generating the training data and for running experiments. The different learning techniques that we use for learning advices are explained in section 5. Next, in section 6 we detail our experiments and analyze the results we obtain. Finally, in section 7 we draw some conclusions and discuss future work.

2. RELATED WORK

Several approaches have been taken to learn to give advice to players in the simulated robotic soccer platforms as a machine learning problem.

Kuhlmann et al. [5] produce three different kinds of advice (formational, offensive, and defensive) before a game from the log files of past games played by a team's opponents. Although they focus on treating advice-giving as a machine learning problem, not all advices are learnt, and thus they combine learnt advices with hand-coded rules. Similarly, Riley et al. [11] consider advice-giving as an action-prediction problem. The learning of advices, in both [11] and [5], are based on the adversary team (by observing previous games played by the adversary to generate the advice) instead of on the team to be coached.

Alternatively, Riley and Veloso focus on adapting advices to the team being coached [9]. They infer a Markov Decision Process (MDP) using observations from agents acting in the environment and solve it to generate advices. Likewise [10], in this paper we focus on adapting advices to the team being coached but unlike Riley and Veloso we use decision tree learning.

In addition to coaching in RoboCup, we find in the literature further contributions to learning advices in other domains. For example, in [10] Riley and Veloso empirically study the coach problem in a predator-prey environment using reinforcement learning.

Decision tree learning has been also used in players learning in the simulated robotic soccer platform. In [12], Stone uses decision trees for pass evaluation. In [4], Konur et al. use C4.5 to learn the action selection strategy when a player holds the ball. In [13], Visser and Weland apply C4.5 to learning aspects of the strategy of a team's adversary. In all these cases the players use a decision tree to learn. However, to the best of our knowledge decision trees have not been used for adapting advices to the team being coached in the simulated robotic soccer platform.

3. COACHING IN ROBOCUP

The simulated robotic soccer platform is a multiagent environment that includes two different teams of eleven agents and an online coach agent for each team that acts as an advice-giving agent [1]. A coach agent aims at improving a team's performance by providing advice to the players. A coach receives a global view of the game, but he does not receive the individual actions or perceptions of the players. He only receives the position and speed of all players and the ball. The coach communicates with the players via a standard coach language called Clang. Clang consists of a set of rules of the type "if condition do/dont {action}", representing advices to players. The *condition* denotes a situation, described by the position of the players and the ball. The directives *do/dont* {*action*} are applicable if the condition is true. The directive *do* {*action*} indicates that the players are recommended to do *action*, whereas the directive *dont* {*action*} indicates that the players are recommended to avoid to do *action*. The players freely decide whether to take into account a coach advices for their decision making during a game or not. In other words, a coach cannot enforce his directives to players.

Clang enables a coach to give advice to players that have been developed by other researchers. The effects of a coach advices depends on how the players exploit them. The same type of advice can have different effects on different players. Ideally, the coach must learn and adapt his advice to the capabilities and limitations of the players he coaches and he must also adapt his advice to the abilities of the opponent team. One important consideration is that a coach cannot change the abilities of the players. He can only exploit their abilities and improve their performance by providing advice to them.

In this paper we focus on a coach that learns advices to improve the passing between teammates. Thus, the coach learns the Clang rules required to improve the number of successful passes between teammates. For this purpose, we want the coach to learn under what conditions he must advice a player either to pass or not the ball to a teammate.

4. SCENARIO

In order to generate training data to learn passing advice and test data to evaluate it we define a scenario composed of two teammates and one opponent. The teammates are coachable players and the opponent is a non-coachable player. Next, we concentrate on learning how an opponent affects the success in passing between two teammates.

In order to generate the training data, our coach sends the "pass always" advice to both teammates. Thereafter, the teammates pass the ball to one another during a whole game while the opponent attempts to get the ball. When a player realizes that he owns the ball, he communicates the coach his intention to pass. When the coach receives this message from a teamate, he generates an example for the training data composed of:

- the absolute position of the passer on the field;
- the distance of the receiver to the passer;
- the distance of the opponent to the passer; and
- whether the pass succeeded or failed.

We only consider two possible outcomes: successful passes (the teammate gets the ball) and failed passes (the opponent gets the ball).

A lost ball situation arises when nobody gets the ball during a fixed period of time after the pass action started. Then, if a teammate gets the ball before fifteen simulation cycles, the coach saves the example as successful. Otherwise, if the opponent gets the ball before fifteen cycles the coach saves the example as failed. Finally, if nobody gets the ball in fifteen cycles the coach discards the example and it is not used in the learning process.

Therefore each example is represented as a tuple:

$$\langle P_x, P_y, R_x, R_y, O_x, O_y, R \rangle \quad (1)$$

Where P_x is the absolute position of the passer on the x axis of the field; P_y is the absolute position of the passer on the y axis of the field; R_x is the distance on the x coordinate of the receiver to the passer; R_y is the distance on the y coordinate of the receiver to the passer; O_x is the distance on the x coordinate of the opponent to the passer; O_y is the distance on the x coordinate of the opponent to the passer; and R is the result.

Attribute R takes on a boolean value that indicates whether the pass succeeded or failed. The other attributes take on real values as follows: $P_x \in [-52.5, 52.5]$; $P_y \in [-34, 34]$; $R_x \in [-105, 105]$; $R_y \in [-68, 68]$; $O_x \in [-105, 105]$; and $O_y \in [-68, 68]$.

Notice that the possible values of the attributes of the passer, P_x and P_y, are the same than the coordinates of the field

along the x and y axes. The possible values of the attributes of the receiver and opponent, R_x, R_y, O_x and O_y, are twice as much as the coordinates of the field because they represent the distance to the passer.

5. TECHNIQUES

Once generated the training data, the coach uses it for learning the advice to pass. We use decision trees to learn advices. Since Clang advices are rules of the form "if condition do/dont action", it is straightforward further on to transform a decision tree into a set of rules Clang.

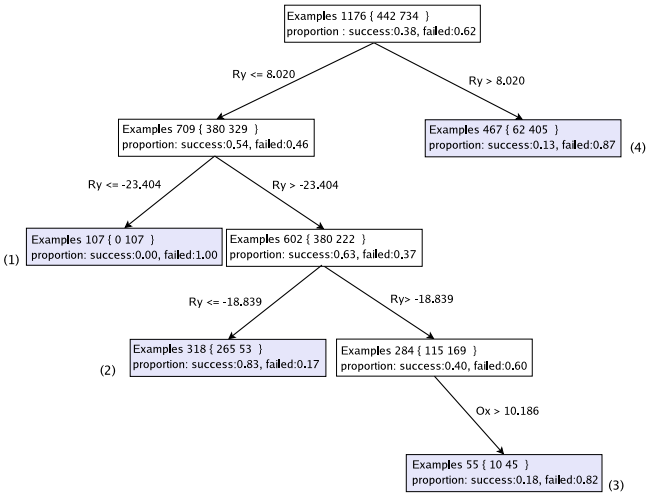

Figure 1: Example of a resulting decision tree

Figure 1 shows an example of a decision tree learned by the coach. The coach generates as many advices as leafs has the tree. Each leaf node of the tree represents an action. We only consider two type of actions, namely "do pass" and "dont pass" because we aim at generating two types of advice: whether to pass or not.

Each leaf of the node can have two types of examples: successful passes and failed passes. Let parameter $\gamma \in [0, 1]$ be a threshold. When the ratio of successful examples in a leaf is greater than γ the leaf will be considered as a leaf that characterizes successful passes. Otherwise, a leaf with a success ratio smaller than $1 - \gamma$ will be considered as a leaf that characterizes failure passes. In our experiments we have used $\gamma = 0.7$ because we consider that this threshold sufficiently characterizes a successful pass ratio. The leafs with a success ratio greater than γ will be converted to some advice of type "do pass", whereas the leafs with a success ratio smaller than $1 - \gamma$ will be converted to an advice of type "dont pass".

The condition of each Clang advice is created through the path from the root to the leaf of the tree. For example, considere figure 2 showing all the rules extracted from the tree in figure 1. Figures 3 and 4 show the second (2) and third (3) rules of figure 2 expressed as Clang advices.

The rule in figure 3 advices player 2 to pass the ball to player 3 when the distance to player 3 on the y axis ranges between -23.4 and -18.84. The rule in figure 4 advices player 2 not to pass the ball to player 3 when the distance to any opponent along the x axis is greater than 10.19 and the distance to player 3 on the y axis ranges between -18.84 and 8.02.

```
(1)  if [ (Ry =< -23.404)] then  FAILED  (1.0 of 107 examples)
(2)  if [ (Ry > -23.404) (Ry =< -18.839)] then SUCCESS (0.83 of 318 examples)
(3)  if [ (Ox > 10.186) (By =< 8.020) (Ry > -18.839)] then FAILED (0.82 of 55 examples)
(4)  if [ (Ry > 8.020)] then  FAILED (0.87 of 467 examples)
```

Figure 2: Rules extracted from the tree in figure 1.

```
(define (definerule Rule_2A direc (
(and
      (and (bowner our { 2 }))
      (and (ppos our { 3 } 1 11  (rec (((pt our 2)+(pt 100.00 -23.40))) (((pt our 2)+(pt -100.00 -18.84))))))
)
(do our { 2 } (pass { 3 })))))
```

Figure 3: Rule (2) of figure 2 in Clang language.

```
(define (definerule Rule_3A direc (
 (and
      (and (bowner our { 2 }))
      (and (ppos opp { 0 } 1 11 (rec (((pt our 2)+(pt 100.00 -100.00))) (((pt our 2)+(pt 10.19 100.00))))))
      (and (ppos our { 3 } 1 11  (rec (((pt our 2)+(pt 100.00 -18.84)))  (((pt our 2)+(pt -100.00 8.02))))))
 )
 (dont our { 2 } (pass { 3 })))))
```

Figure 4: Rule (3) of figure 2 in Clang language

In order to build a decision tree we experimented with different heuristic measures to compare which one leads to best results. In particular we used the Gain criterion [7], the GainRatio criterion [7] , and the Mantaras normalized Distance [6] because these are the most popular measures.

We have also experimented with three different learning techniques to build the decision tree in order to compare which technique leads to best results. The three techniques are represented in figure 5, and they are called (1) Global technique, (2) Local technique and (3) Pruned technique.

In the Global technique we use discretization for the attributes of the training examples that are continuous. The Global technique has four steps: Discretization, Decision tree induction, Branch elimination and Rule generation. As shown on (1) in figure 5, the first step is to perform a discretization of each continuous attribute before building the tree. Once the data are discretized, we induce the decision tree with the aid of an ID3 algorithm [8]. Before translating the resulting decision tree into advices, we perform a pruning process, named Branch elimination, that generates a new tree. The Branch elimination step consists on eliminating those branches of the tree whose leaves have a ratio of successful passes between γ and $1 - \gamma$ (0.7 and 0.3). We name trimmed tree the tree generated through the Branch elimination step. The last step is the Rule generation, which consists on translating every branche of the trimmed tree into advices. Thus, we only consider those situations where the pass turns out to be either good or bad. In both cases, the condition of the advice is constructed through the path from the root to the leaf. Each node of the path is an element of the advice, and all elements are combined with an "and" operator.

In the Local technique (see (2) on figure 5) we create the decision tree from the continuous data in the training exam-

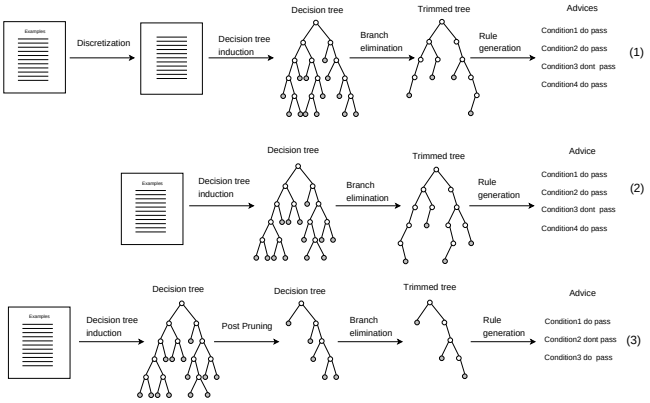

Figure 5: Learning techniques

ples. The Local technique has three steps: Decision tree induction, Branch elimination, and Rule generation. Whereas the Decision tree induction is different from the Global technique, both the Branch elimination and the Rule generation steps are the same than those in the Global technique. The discretization is carried out through the decision tree attribute selection step. At each step of the decision tree attribute selection we decide a split value for each attribute. Thus, we obtain a partition of the training examples into two parts for each attribute. The attribute that is selected is the one with the best partition of examples according to the heuristic measure criterion. This process is implemented at each node of the tree. Therefore, as the construction of the decision tree progresses the number of examples is smaller and the discretization is only based on these examples. The algorithm process stops when the number of examples in the node is less than 5% of the examples in the root node. Once built, the steps to convert the decision tree into advices are the same as those employed by the first technique, namely Branch elimination and Rule generation.

The Pruned technique (see (3) on figure 5) is similar to the Local, but before performing the Branch elimination, we use a post-pruning method. The Pruned technique has four steps: Decision tree induction, Post-pruning, Branch elimination, and Rule generation. The Decision tree induction step, the Branch elimination step and, the Rule generation step are the same as in the Local technique.

The Post-pruning step is based on pruning the subtrees of the nodes of the decision tree that satisfy either equation 2 or equation 3:

$$ratio(n) \geq \gamma \quad \wedge \quad \forall l \in leaf(n) : ratio(l) \geq 1 - \gamma \qquad (2)$$

$$ratio(n) \leq 1 - \gamma \quad \wedge \quad \forall l \in leaf(n) : ratio(l) \leq \gamma \qquad (3)$$

A node n of the tree satisfies equation 2 if the node has a success ratio greater or equal than γ and all their subtree leafs have a success ratio greater or equal than $1-\gamma$. While a node n satisfies equation 3 if the node has a success ratio less or equal than $1-\gamma$ and all their subtree leafs have a success ratio less or equal than γ.

With the Post-pruning step we expect generate less rules and less conditions in the rules without lost of accuracy. For example, in our experiments we prune the subtree of a node n that has a success ratio greater or equal than 0.7 and that all their subtree leafs have a success ratio greater or equal than 0.3. We prune the subtree of the node n because if we did not prune its subtree, all branches of its subtree either will be eliminated through the Branch elimination step (because its leaves' success ratio range between 0.3 and 0.7) or will remain in the trimmed tree (because its leaves' success ratio is greater than 0.7, like the node). As to the latter case, the leafs in the subtree would generate the very same type of advice ("do pass") than the node n, which makes such advice redundant.

6. EMPIRICAL EVALUATION

In order to validate our hypothesis stating that learning advices help significantly increase the number of successful passes between teammates we run two of experiments. In the first experiment, Experiment 1, the coach sends to the players the "do/dont pass" learned advice. In the second experiment, Experiment 2, the coach only sends to the players the "dont pass" learned advice.

To run our experiments we employ the Wyverns soccer team players [3], developed by Patrick Riley. In order to use Wyverns' players in the scenario explained in section 4 we needed to change the way players communicate their intention to pass to the coach. Then, when a player intends to make a pass, he sends a message to the coach to communicate his intention so that the coach can accordingly create a new example. Therefore, notice that we do not make any extensions to the behavior of players; we only add a new communication action with the coach. On the other hand, we use the UvA Trilearn 2003 soccer simulation team [2] as opponent to Wyverns because its players play good enough and do not need a coach to play.

The first step is generating the training data. We launch all players, namely two teammates and one opponent, to play in the scenario described in section 4. We also launch our coach, that sends to the teammates the "pass always" advice. In the training scenario, our coach generated a total of 1176 examples in one game. These examples have a ratio of successful passes of 38%. Once the coach has generated the training data, we use these examples to run the three learning techniques explained above (see section 5). We consider that the 1176 examples generated during the game are enough. We run each learning technique three times, one for a each heuristic measure (Gain criterion, GainRatio criterion and Mantaras normalized Distance). Altogether we perform nine different runs to learn advices. As a result, we obtain nine different trees along with a set of advices per tree.

In table 1 we show the percent of correctly classified examples by both the decision tree and the trimmed tree over a total of 1176 examples. The Measure/Technique column lists each combination of technique with heuristic measure. The DTree column lists the accuracy of the decision tree obtained before applying branch elimination. The Trimmed tree column lists the accuracy of the trimmed tree obtained with the Branch elimination step (the tree used for building the advice rules). The Trimmed tree column also shows, in

parentheses, the percentage of examples that the trimmed tree classified. Notice that while the decision tree is able to classify all training examples the trimmed tree is not.

Table 1: Percentage of correctly classified instances of training data for the decision tree and for the trimmed tree.

Measure / technique	DTree	Trimmed tree
Gain		
Global	81.5	86.8 (79%)
Local	82.4	86.8 (84%)
Pruned	82.4	84.8 (90%)
GainRatio		
Global	81	81 (100%)
Local	80.4	82 (93%)
Pruned	80.4	80.4 (100%)
Mantaras Distance		
Global	81	87 (76%)
Local	82.7	89.3 (74%)
Pruned	82.7	86.8 (81%)

The accuracy of the DTree is similar for the three techniques and for the three heuristic measures (around 80%). As to the Trimmed tree, we observe an increment of accuracy as well as a decrement of the percentage of classified examples compared with respect to DTree. Notice also that the measure that classifies more examples is Gain Ratio, and the Pruned technique is the one that classifies more examples.

We have also compared the nine different sets of advices by evaluating the effects of each advice on the players' performance when passing the ball. For testing purposes, we run two experiments where the coach sends the advice generated by the tree to the Wyverns' teammates. The coach uses the same scenario used for generating the training data (see section 4) to evaluate the advice. A test consists on launching the two teammates and one opponent and also launching our coach to send to the teammates the learned advice. We have run a test for each combination of technique and heuristic measure. Each test finishes when 1000 passes are make by the teammates.

The players use the received advice to help them decide their actions. Notice that since not all situations are covered by the advice the time needed to generate the very same number of examples can be greater in a test scenario than in the training scenario. At testing time, there are situations where one of the teammates owns the ball for which he does not have any advice, and however he has to choose the action to do, (either "pass to teammate" or some other action). These are situations that are not covered by the advice. Among the situations covered by the advice, there are situations where one of the players owns the ball and the player has to consider to do the pass after receiving a "do pass"; and other situations where the advice is "dont pass" and he has to consider which other action to choose.

In table 2 we show the success ratio of the nine different sets of advices in 1000 passes after sending the learned advices (Experiment 1). We need now to compare these values with 38% of successful passes of the players obtained in the baseline scenario, when the coach does not use any learning advice.

The results show that only the Global technique with the Mantaras Distance measure is not statistically significant worse than the 38% of successful passes without learning. The other results present a statistically significant improvement of percentage of successful passes with respect to the 38% of successful passes without learning; they improve the percentage of successful passes around 15%. In particular, the measure producing the best results is the Gain, whereas the measure leading to the worst results is the Mantaras Distance. The technique producing the worst results is Global, whereas Local technique produces the best results. The combination showing best results is composed of the Local technique along with the Gain Ratio measure, improving 19% the percentage of successful passes.

Table 2: Ratio of successful passes in Experiment 1. (38% baseline)

Measure	Global	Local	Pruned
Gain	54.7	55.4	55.5
GainRatio	47.5	57	54.1
Mantaras Distance	35.5	52.7	47.6

The results show that the learnt advices improve the percentage of successful passes. However, we have observed that while in the training scenario the players only need one game to generate 1176 examples, in the test scenario the players need several games to generate 1000 examples. This creates a time overhead to generate the same number of examples for the test scenario. This time overhead is motivated by the fact that in the training scenario the sent advice was "pass always", which covers all possible situations; whereas in the test scenario the learning advice does not cover all possible situations. Moreover, in the test scenario there are situations where the advice rule is "dont pass". In particular, the Local technique with the Gain Ratio measure is the combination that needs more games to generate 1000 examples in the test scenario. It is also the combination that shows best success ratio results.

In order to reduce the time to generate test examples, we have run Experiment 2. In Experiment 2, we have evaluated the effects of the coach only sending the learnt "dont pass" advices, and the "do pass" advice is simply "pass always". Thus, the player is always enforced to pass except in those situations where the coach advices against it. In other words, the coach only advices against the bad situations where the pass must be avoided and, thus, increase the probability to pass between teammates and decrease the time to generate the examples. Since the technique producing the worst results in Experiment 1 is Global we have not evaluated this technique in Experiment 2. Table 3 shows the success ratio of the six different sets of advices in 1000 passes after sending only the "dont pass" advice (Experiment 2). The results show that in general the ratio of successful passes improves considerably over 38% (baseline). Although the improvement is less significant than in Experiment 1, the reduction of the time needed to generate 1000 examples is remarkable. In this experiment, the measure leading to best results is the Gain Ratio. The combination that shows best results is the one composed of the Local

technique and the Gain Ratio measure, improving 16% the percent of successful passes. While the Pruned technique with the Mantaras Distance measure improves the percentage of successful passes 16% and the Pruned technique with the Gain Ratio Measure improves 15%.

Table 3: Ratio of successful passes in Experiment 2. (38% baseline)

Measure	Local	Pruned
Gain	50.8	47.4
GainRatio	54.8	53.1
Mantaras Distance	36.9	54.7

7. CONCLUSIONS AND FUTURE WORK

In this paper we have explored how coaching can help to improve the performance of RoboCup players. We have presented a coaching approach based on learning how to advice the playing agents. We have employed decision trees to learn advices from observing the actions of the agents to be adviced. We have proposed three different learning techniques to create the advice and have evaluated each technique with the Gain, Gain Ratio and Mantaras Distance measures.

We have empirically validated our hypothesis stating that learning advices help significantly increase the number of successful passes. In the baseline scenario, when the coach does not use any learning advice, the ratio of successful passes is 38%. We have run two experiments and we have observed that the success in passing the ball significantly improves when exploiting learnt advices. Thus, the learnt advice sent by the coach increases between 9% and 19% the number of successful passes made by players over the 38%.

Concerning what learning technique is better for learning advices, the results show that there is not a clear winner. The Pruned technique has worse results than the Local technique, but it generates less amount of advice, and thus less rules to send to players. Not all combinations of heuristic measures and techniques have the same results. The combinations that lead to the best results depends on the type of experiment. In Experiment 1, when the coach sends the learning advice to the players, the measure Gain has good results with the three techniques, whereas the measure Gain Ratio has good results with the Local and Pruned techniques. In Experiment 2, when the coach only sends the "dont pass" advice learned, the measure Mantaras Distance has good results with the Pruned technique and Gain Ratio with both techniques, Local and Pruned.

Notice that the experiments have been only performed with one team, [3], and these final conclusions of witch combination is better are based only for these players. We plan to perform the same experiments with other players to compare the results between different players. In the future we plan to extend the learning to further scenarios with more teammates and more opponents. We also consider to apply our learning methods to other actions, not only to passes. These extensions can help us evaluate experiments in a real game.

8. REFERENCES

[1] Robocup soccer server manual for soccer server version 7.07 and later, 2003. http://sourceforge.net/projects/sserver.

[2] Uva trilearn 2003 soccer simulation team, http://staff.science.uva.nl/ jellekok/robocup/2003.

[3] Wyverns soccer team source code, http://www.cs.cmu.edu/ pfr/thesis/wyverns.tar.bz2.

[4] S. Konur, A. Ferrein, and G. Lakemeyer. Learning decision trees for action selection in soccer agents. In *Proc. of Workshop on Agents in dynamic and real-time environments*, 2004.

[5] G. Kuhlmann, P. Stone, and J. Lallinger. The UT Austin Villa 2003 champion simulator coach: A machine learning approach. In D. Nardi, M. Riedmiller, and C. Sammut, editors, *RoboCup-2004: Robot Soccer World Cup VIII*, pages 636–644. Springer Verlag, Berlin, 2005.

[6] R. L. D. Mántaras. A distance-based attribute selection measure for decision tree induction. *Mach. Learn.*, 6(1):81–92, 1991.

[7] J. Quinlan. *C4.5: Programs for machine learning.* 1993.

[8] J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106.

[9] P. Riley and M. Veloso. Advice generation from observed execution: Abstract Markov decision process learning. In *aaai2004*, 2004.

[10] P. Riley and M. Veloso. Coaching advice and adaptation. In D. Polani, A. Bonarini, B. Browning, and K. Yoshida, editors, *RoboCup-2003: The Sixth RoboCup Competitions and Conferences.* Springer Verlag, Berlin, 2004.

[11] P. Riley, M. Veloso, and G. Kaminka. An empirical study of coaching. In H. Asama, T. Arai, T. Fukuda, and T. Hasegawa, editors, *Distributed Autonomous Robotic Systems 5*, pages 215–224. Springer-Verlag, 2002.

[12] P. Stone and M. Veloso. A layered approach to learning client behaviors in the RoboCup soccer server. *Applied Artificial Intelligence*, 12:165–188, 1998.

[13] U. Visser and H.-G. Weland. Using online learning to analyze the opponent's behavior. In *RoboCup*, pages 78–93, 2002.