Using Online Learning to Analyze the Opponents Behavior

Ubbo Visser and Hans-Georg Weland

TZI - Center for Computing Technologies, University of Bremen
Universittsallee 21-23, D-28334 Bremen, Germany
{visser|weland}@tzi.de
http://www.virtualwerder.de/

Abstract. Analyzing opponent teams has been established within the simulation league for a number of years. However, most of the analyzing methods are only available off-line. Last year we introduced a new idea which uses a time series-based decision tree induction to generate rules on-line. This paper follows that idea and introduces the approach in detail. We implemented this approach as a library function and are therefore able to use on-line coaches of various teams in order to test the method. The tests are based on two 'models': (a) the behavior of a goalkeeper, and (b) the pass behavior of the opponent players. The approach generates propositional rules (first rules after 1000 cycles) which have to be pruned and interpreted in order to use this new knowledge for one's own team. We discuss the outcome of the tests in detail and conclude that on-line learning despite of the lack of time is not only possible but can become an effective method for one's own team.

1 Introduction

The standard coach language provides the on-line coaches in the RoboCup soccer simulation league with a possibility to rapidly change the behavior of his team. In addition, its existence allows a competition between coaches. In order to achieve a successful coaching, a lot of information about the opponent has to be collected, to which the coach can react and change his own team according to the opponent. For this reason several methods have been introduced in the past to analyze on-line and to adapt to the opponents. These papers demonstrated how to recognize team formation [Visser et al., 2001] or how to adapt to play situations [Riley and Veloso, 2002]. [Drücker et al., 2002] showed the idea and a first prototype of the method presented in this paper.

Very important aspects of the behavior of a soccer team are the goalkeeper and the pass behavior.

A pass is a frequent event within a soccer game. It allows the team passing to move the ball across a great distance in a short time and to defeat the opposing defenders. Thus, successful passing can be a great advantage. On the other hand, a pass is always a risk. When the ball is passing it moves without being guarded by a team member who could change the direction of the ball, if necessary. This gives the opposing players the possibility to intercept the pass. These intercepted passes are a certain disadvantage. When a team is passing less successful in certain situations than in others, it makes sense for the opposing team to try to create these situations as often as possible. However, in order to do this, it must be known which factors lead to these mistakes.

Therefore, analyzing the goalkeeper the moment when he leaves the goal is of special interest. A possibility to play around him or to reach a point near the goal while he does not attack the forwards, produces great chances to score. It is important to know which factors have led to the players' decision. Also, it is important to know the threshold where they react in a certain way. Thus, it is crucial to use a method that is not hampered by pre-discreeted values but does the discreetization itself. As the outcome of such a method should directly be used by the on-line coach, the results require a certain form. They should be employed to generate instructions for the players. For these constraints the time series-based decision tree induction described in [Boronowsky, 2001] seems suitable. As this method operates with continuous-valued time series, a pre-discreetization of the data is not required. The method finds the split points and therefore the thresholds where the analyzed player act.

2 Time series-based decision tree induction

This method consists of a entropy minimization heuristic that only has to be calculated for certain points as opposed to C4.5[Quinlan, 1993] where all possible split points are calculated. This is an important advantage to other decision tree algorithms. Due to the great amount of possible split points, the calculation effort can be very high with continuous data.

The basic ideas for the optimization of the method we use are similar to those of ID3 and C4.5 described in [Fayyad and Irani, 1992]. It shows that the entropy minimization heuristic can only be minimal at the so-called boundary points. A boundary point is a point within the range of a certain attribute whose neighbors belong to two different classes. Thus, the range of an attribute holds for a boundary point T, and

$$A(e_1) < T < A(e_2). \tag{1}$$

$E_1, e_2 \in S$ belong to the set of all samples S, $A(e)$ is the attribute value of e and

$$\neg\exists e_3 \in S : A(e_1) < A(e_3) < A(e_2) \tag{2}$$

where e_1 and e_2 belong to disjunctive classes.

This approach already leads to a noticeable increase in efficiency given appropriate samples. If the samples include overlapping classes the increase of efficiency ends. There are boundary points in the overlapping area in such cases. The calculation then has to be done for all these points because the entropy minimization heuristic can be minimal for all of them. In RoboCup environments such overlapping areas cannot be ruled out.

It can be shown though that for continuous equal distributions the entropy minimization heuristic can only be minimal at so-called true boundary points. True boundary points are those boundary points that are located at the boundaries of the interval of one class assignment. When a continuous equal distribution can be reached, the number of interesting points can be reduced and therefore the calculations. This leads to an significant increase in efficiency.

Since this decision tree induction does not work on a set of examples but on time series, the calculation of the optimal split point does not depend on the amount of examples belonging to a certain class. Instead, it depends on the time interval of such an assignment. Instead of $freq(C_j, S)$ and $|S|$ the interesting issues are

- the duration of the assignment of the time series smaller than the split point y to a class C_j $time_{C_j}(y)$,
- the duration of the assignment of the time series above the split point $\overline{time_{C_j}}(y)$, and
- the total duration of the assignment to one class $tmax_{C_j}$.

Thus, there isn't an assignment to a class for every single sample, but there is a qualitative abstraction to one or a combination of multiple time series. This defines which time intervals are assigned to which class. Figure 1 shows an example of a qualitative abstraction of a time series. It shows the behavior of a goalkeeper. At time t_{begin} the distance between the keeper and the goal remains stable. At time t_1 the distance increases which means that he leaves the goal. This state holds until time t_2. The function will be abstracted to class B_{stays} and C_{leaves} accordingly.

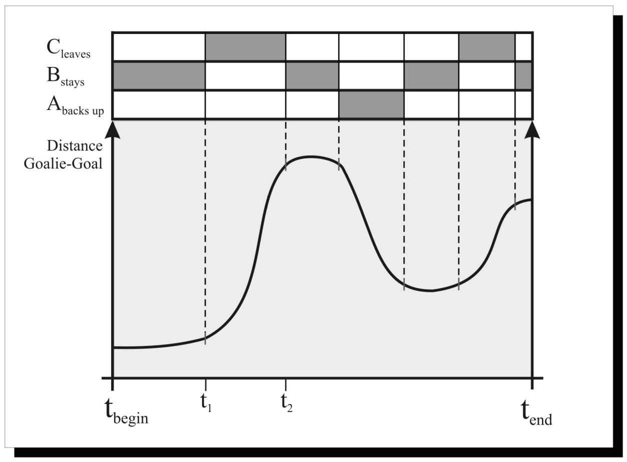


Fig. 1. Qualitative abstraction of the behavior of a goalkeeper

For partial linear measured value gradients it can be shown that the entropy minimization heuristic can be calculated efficiently [Boronowsky, 2001]. This is based on the characteristic that the duration of a class assignment $time_c(y)$ for partial linear measured value gradients can be described by linear functions. Such linear durations of class assignments are continuously equally distributed. Thus, the correlation between a continuous equal distribution and an entropy minimization heuristic, as described above, can be used.

It is therefore possible to perform an efficient decision tree induction for continuous valued time series, if these time series consist of partial linear functions. As it cannot be assumed that such partial linear measured value gradients are found, they must be adapted in an appropriate way. This is done by an approximation of the continuous measured values by partial linear functions. As an approximation does not exactly equal the original function an approximation error occurs. This error should be as small as possible, especially at the split points. In general, one can achieve a better approximation by increasing the number of linear functions. On the other hand, this leads to a higher number of potential split points. Thus, the reduction of the approximation error leads to a loss of efficiency. Although, certain points seems to be very important and should therefore be used as boundary points for the linear functions. These are

- the start and end of the used time series
- the times of a change in the class assignment
- the extreme values of the measurement course

It makes sense to use the start and end of the time series because this is the only way to represent the whole series. By using the points where the class assignment changes as boundary point for the linear approximations, a linear function is always assigned to exactly one class. This makes the calculations of the duration of class assignments easier. The extreme values are interesting because they give the option to find a split point, which separates the time series in such a way that the complete series assigned to one class is under or above this split point.

With these partial linear functions the entropy minimization heuristic only need to be calculated for the boundary points of the linear functions because it can only be minimal at these points. Fig.2 shows an approximation with four boundary points (1-4) and the according four points for potential horizontal splitting (y1-y4). To calculate the entropy, the information contents of the duration of class assignments and ($info(y)$) an above ($\overline{info}(y)$) the potential split point has to be calculated.

$$info(y) = -\sum_{i \in C} \frac{time_i(y)}{\sum_{k \in C} time_k(y)} ln \left(\frac{time_i(y)}{\sum_{k \in C} time_k(y)} \right) \tag{3}$$

$\overline{info}(y)$ has to be calculated in the analogue, by changing $time(y)$ to $\overline{time}(y)$. The entropy can be calculated by

$$entropy = \frac{\sum_{i \in C} time_i(y) info(y) + \sum_{i \in C} \overline{time_i}(y) \overline{info}(y)}{\sum_{i \in C} tmax_i} \quad (4)$$

This calculation must be done for all boundary points in all time series. At the point where the results are minimal the according time series is split horizontally. All the others are split vertically at certain points. These points are those where the horizontally splitted time series crosses the value by which it is split. The result is that two new time series are created for every existing time series. One consists of those time intervals in which the horizontal splitted series is greater than the split threshold and the other of those where it is not. This process is then repeated with the two new sets until a certain end criterion is reached.

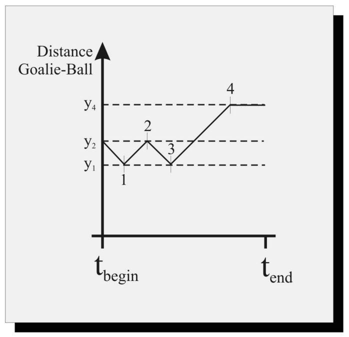


Fig. 2. Partial linear approximation with potential points for horizontal splitting

The ending criterion cannot be the number of correct classified examples as used in regular supervised symbolic machine learning techniques. As it is a time series-based method the ending criterion should be the correct classified time intervals.

At the splitting of the time series the approximation error has to be considered. If the splitting is only calculated by the approximations and not on the basis of real functions there can be errors in the horizontal as well as in the vertical splitting. At the horizontal splitting the approximation can be different from the real value on the time axis and at the vertical there can occur an error on the ordinate. This can be prevented by adapting the approximation in a suitable way. Therefore, all points are added to the approximation that should be split horizontally where the real functions crosses the threshold y. To the other approximations new points are added at all vertical split points.

3 Preprocessing of the used data

In order to use the method in an online-coach it has to be defined what should be analyzed and which data should be used for it. In this paper we focus on two scenarios:

1. We analyze the moment when the goalkeeper leaves the goal.
2. We analyze the pass behavior of opponent players

In order to do this a suitable qualitative abstraction must be found. It defines which time intervals are assigned to which class, e.g. if the goalkeeper leaves the goal or if he stays on the line. This leads to the problem that the things that should be learned are not always directly included in the data provided by the soccer server. Therefore, the given data have to be combined or even an element of guessing has to be included. It is important to note that we can only analyze what is happening and that we cannot recognize the player's intention.

The result of this method always depends on the abstraction. If the abstraction is not correct something different is learned. A slight error in the abstraction of the goalkeeper could lead to a tree that has learned the movement of the goalkeeper rather than when he is leaving the goal. These results cannot be used correctly to improve the behavior of a team.

In order to use a decision tree algorithm it is necessary to choose suitable attributes from which the tree is built. It is essential that these attributes represent those values that are important for the decisions of the opponent player. If they are not represented in the attributes the behavior cannot be learned.

3.1 Analysing Goalkeeper Behavior

In the analysis of the goalkeeper's behavior the moment when he leaves the goal is of special interest, because this is when the goal becomes vacant. Thus, a qualitative abstraction is chosen which represents this behavior. The movements of the goalkeeper are represented in three classes, *(a) goalkeeper stays in goal, (b) goalkeeper leaves goal and (c) goalkeeper returns to goal*. This calculation is based upon the movement of the goalkeeper towards the ball. If he moves towards the ball he leaves the goal, and if he goes away from the ball he returns to the goal. In all other cases he stays in the goal. This is not always correct in relation to the goal, but it represents what should be learned. The movement vector of the goalkeeper and the position of the ball are used to compute the abstraction. The length of the movement vector gives the speed, and the angle between the movement vector and a vector from the goalie to the ball are used for the direction.

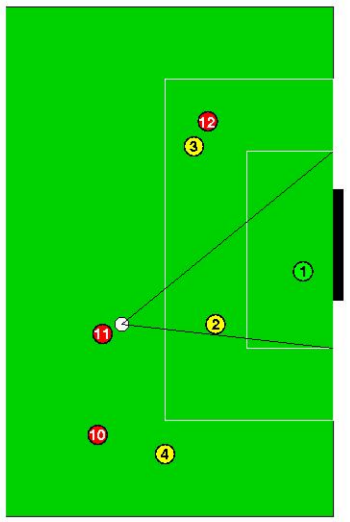


Fig. 3. Cone from ball to both sides of the goal and some positions of opponent players (1-3) as attributes for the learning process

As described above, suitable attributes must be chosen. These attributes should include those that are used by the analyzed goalkeeper to make his decisions. The position of the ball seems to be very important. It will be the major aspect in the goalkeeper's decisions.

Murray [Murray, 1999] even describes a goalkeeper whose decisions are totally based upon the ball position. Because of the relative view of the goalie and the abilities of our own players, the relative distance of the ball to the goal and the goalkeeper seems more interesting than its absolute position. Another interesting fact are the positions of the other players (see figure 3). Especially those who can directly interfere in a possible shot at the goal. Particularly the players within the penalty area are relevant to the goalkeeper because of his ability to catch the ball. This is why he may react differently to players in this area. Thus, the number of forwards and defenders in the penalty area is used to analyze the goalkeeper.

Very important for a goalkeeper is the question whether the opposing ball carrier can run and shoot at the goal without being attacked by a defender. In these cases a goalkeeper should leave the goal and run towards the ball carrier to prevent him from shooting into a free corner of the goal. Hence, the defenders within a cone from the ball to the sides of the goal (figure 3), which are likely to intercept the ball, are used as an attribute for the decision tree.

3.2 Analysing Passes

In order to analyze the pass behavior, a qualitative abstraction and some input values have to be found. This leads to some problems owing to the kind of the present data. The coach only knows the positions and movements of all players and of the ball, but not whether a player kicks the ball or not and in which direction the ball was kicked. Therefore, it is impossible to see in the data given by the soccer server who kicked the ball or even if the ball was passed at all.

However, because there are certain rules describing the changes in speed and direction of the ball, it can be calculated from the movement of the ball whether it was kicked or not. Without being kicked by a player the ball cannot gather speed. The same applies to a sudden stop. In both cases a player must have sent a kick command.

This does not apply to a change of direction. A direction change could also happen as a result of a collision with a player. Thus, it cannot be verified that the ball was kicked when it changes its direction. A bigger problem is the question who kicked the ball. Suppose the ball is within the kickable area of one player and one of the events described above happens at the same time. Then, only that player can be the one who kicked the ball. But if the ball is in the kickable area of several players it cannot be determined who kicked it. It also is not possible to tell which intention the player had when he kicked the not be told. On the other hand, it makes no difference to a team whether it gets the ball by intercepting a pass, by dribbling, or by a goal shot. If a team can provoke situations that lead to an interception always has an advantage.

These considerations lead to a qualitative abstraction based upon two kick events. Every kick event is compared with the prior one. It is important which players did the two kicks. The qualitative abstraction assigns these events into four classes:

- pass between two opponents,
- opponent dribbles,
- pass of an opponent was intercepted, and
- team-mate kicked the ball.

A kick command is supposed to have happened if the ball gathers speed or is suddenly stopped. The player who is next to the ball is assumed to have kicked.

When defining which data should be used as attributes for the learning method one has to take into account that the method has to find out what the player does. Therefore, it is important to use the values used by the player while making decisions. The surroundings of the ball carrier are of special significance for the analysis of pass behavior, especially the distances and directions of the nearest players (see figure 4). It makes no sense to use absolute positions because the player only gets local information and is likely to make his decisions based on relative values, as given to him by the server. Also, players far away from the ball carrier are of no importance. The ball carrier does not see them well, maybe not at all, and a pass across a very long distance is a high risk. Thus it is unlikely that such players play an important part in the decisions of the passer.

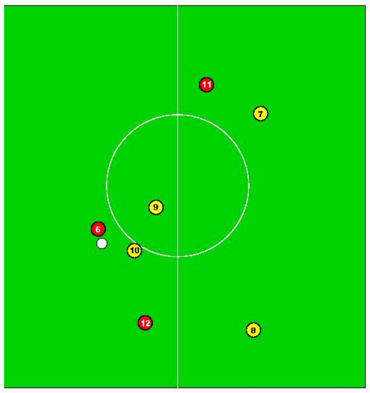


Fig. 4. A passing situation, attributes are positions of the opponent players, surrounding of the ball carrier (e.g. distance and directions of nearest player)

While calculating the angles to the other players, it has to be taken into account that in soccer it is more important whether a player is located towards the middle of the field or towards the sideline than to the right or to the left. If e.g. the ball carrier is attacked from his left side it has a different meaning to him whether he is on the right or on the left side of the field. If he is on the right the defender blocks his way to the goal and pushes him towards the sideline. On the other hand, if he is on the left side the defender attacks him from the sideline and the way towards the goal is free. This applies even more in the simulation league because there are only physical differences between the players if a team uses the optional heterogeneous players. And all players can kick the ball to the left as well as to the right. This is why all angles on one half of the field are flipped horizontally. As a result, all passes to the middle have a similar angle and can be represented by the decision tree in the same branch.

Another aspect is the horizontal position of the passing player. The pass behavior may change depending on how far the ball carrier is away from his goal line. E.g. a pass played directly in front of the own goal carries a high risk of being intercepted, thus leading to a good scoring position for the opponent, while a pass in front of the opposing goal may give team-mates the chance to score. Additionally, some parts of a team may pass with a higher risk than others. VirtualWerder 00/01 does take this differences into account. The defenders play as securely as possible while the forwards sometimes even pass into the penalty area without knowing whether anybody is there.

In both cases, goalkeeper and pass behavior, the decision tree algorithm is used every 1000 cycles to generate rules about the opponent's behavior.

4 Results

To test the quality of the method several test games were played with different teams of last year's competition. VirtualWerder, Robolog Koblenz and Karlsruhe Brainstormers were used to evaluate the implemented algorithm. Robolog was used because, in contrary to most other teams, it is logic- based. The were chosen because they represent a reactive team and because they finished second in the last competition.For technical reasons, Tsinghuaeolus, the world champion, could not be used. It is a Windows-based team while the computers available for the tests where Linux computers. The binaries of FC Portugal were not available at the time of the tests.

4.1 Goalkeeper

To analyze the goalkeeper a qualitative abstraction was used based upon the movement of the goalkeeper, as described above, and six time series. Here are the time series:

- Series 0 - distance ball goalkeeper
- Series 1 - speed of the ball
- Series 2 - distance ball goal
- Series 3 - number of defenders within the penalty area
- Series 4 - number of forwards within the penalty area
- Series 5 - number of defenders that may intercept a direct shot at the goal

After 1000 cycles the decision tree is computed for the first time, based on data shown in table 1.

Table 1. Typical input data for the learning algorithm w.r.t. a goalkeeper

	Series 0	Series 1	Series 2	Series 3	Series 4	Series 5	Class
1	50	0	52.5	1	0	3	0
2	50	0	52.5	1	0	3	0
3	50	0	52.5	1	0	3	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1000	88.1415	0.487918	90.3164	1	0	0	1

For the tests with respect to the analysis of the goalkeeper ten test games where played with

– Robolog vs Brainstormers
– Robolog vs VirtualWerder

```
if   series 4 < 2.000000
and  series 0 < 20.105579
then 0(1) 1(0) 2(0)

if   series 4 < 2.000000
and  series 0 > 20.105579
then 0(0.71219) 1(0.245547) 2(0.0422634)

if   series 4 > 2.000000
and  series 0 < 0.385009
then 2(0.950617) 0(0.0493827) 1(0)

if   series 4 > 2.000000
and  series 0 > 0.385009
then 2(0.48) 0(0.34) 1(0.18)
```

Fig. 5. The Brainstormers goalkeeper in a game against Robolog

```
(1) if   series 3 < 1.000000
    then 0(1) 1(0) 2(0)

(2) if   series 3 > 1.000000
    and  series 3 < 4.000000
    and  series 1 < 0.031605
    and  series 0 < 6.774502
    and  series 0 < 5.984714
    then 0(1) 1(0) 2(0)

(3) if   series 3 > 1.000000
    and  series 3 < 4.000000
    and  series 1 < 0.031605
    and  series 0 < 6.774502
    and  series 0 > 5.984714
    then 1(0.8) 0(0.2) 2(0)

(...)

(4) if   series 3 > 1.000000
    and  series 3 > 4.000000
    and  series 2 < 7.680346
    then 1(0.75) 0(0.25) 2(0)

(5) if   series 3 > 1.000000
    and  series 3 > 4.000000
    and  series 2 > 7.680346
    and  series 0 < 6.799984
    then 1(1) 0(0) 2(0)

(...)
```

Fig. 6. The VirtualWerder goalkeeper in a game against Robolog

In the first constellation, both goalkeepers were analyzed, and in the second, for technical reasons, only the one from Virtual Werder.

The rules concerning the Brainstormers' goalkeeper are shown in figure 5. We see that the goalkeeper reacts at different distances to the ball (series 0) depending on the number of attackers in the penalty area (series 4). This change in the distance was often noticed but sometimes it depended on the number of defenders in the penalty area. However, this might be the same because both sounds like a breakaway.

The tests with the VirtualWerder goalkeeper revealed noticeable longer rules than with the two other goalkeepers. One possible explanation is that the VirtualWerder field players cannot keep the ball out of their penalty area and therefore the goalkeeper needs often to react. As this leads to more changes between the classes, he can be better analyzed or he may have a more complex behavior than the others. The rules in figure 6 reflect a part of the behavior of the VirtualW-erder goalkeeper in a game against Robolog. The rules show that the goalkeeper changes his behavior depending on the number of defenders in the penalty area (series 3). The first rule reveals that he stays in the goal if there are no defenders present. While the second and third shows that if there are one to four defenders (rules 2 and 3,) he makes his decision based on the speed of the ball (series 1) and the distance from the ball to him. Having more than four defenders within the penalty area (rules 4 and 5,) the decision is based on the distance of the ball to the goal (series 2).

The decision tree method does not produce interesting results about the goalkeeper if there are not enough scenes where he reacts. This can happen if the opponent is too strong for the team of the coach. It may happen that one team is not able to bring the ball into the opposing penalty area. Sometimes teams even have problems to cross the middle line. In these cases the goalkeeper does not have to act often enough to be analyzed, or maybe he doesn't act at all. In such cases the method only delivers one rule saying that the goalie doesn't move at all. But this is no problem, because if the goalkeeper needs not to act it is not an advantage to know what he would do if he had to. At first the other parts of the play have to be improved. If this is possible fast enough there might be enough information to analyze the goalkeeper later on when there is a possibility to draw an advantage out of this knowledge. This problem was revealed by the tests with VirtualWerder. This team was not able to produce enough pressure on the other two teams to produce enough goalkeeper-scenes. Thus, there were no sensible results about the other teams from these games.

4.2 Pass

According to the reflections in 3.2 ten time series had been chosen as attributes to analyze the pass behavior.

- Series 0 - distance to the next opponent
- Series 1 - angle to the next opponent
- Series 2 - distance to the second next opponent
- Series 3 - angle to the second next opponent
- Series 4 - distance to the next team-mate
- Series 5 - angle to the next team-mate
- Series 6 - distance to the second next team-mate
- Series 7 - angle to the second next team-mate
- Series 8 - side of the passer
- Series 9 - x-position of the ball

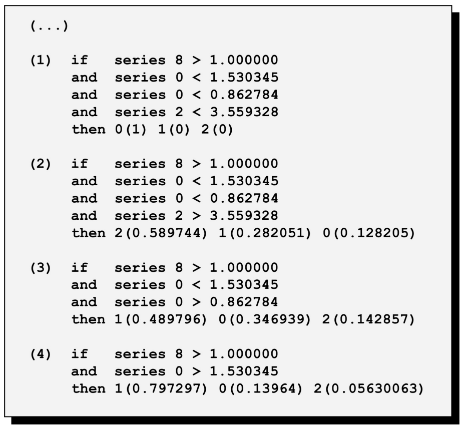
The distances and angles are always relative to the ball carrier because he makes his decisions based on his own perceptions. Because of the reasons described above, the angles are horizontally flipped in one half of the field, hence, the angles towards the middle of the field are always negative while the angles towards the sidelines are positive. After 1000 cycles the rules are generated from data shown in table 2.

Table 2. Typical input data for the learning algorithm w.r.t a pass

	Series 0	Series 1	Series 2	Series 3	Series 4	...	Series 9	Class
1	9.01388	-2.33172	10.5	-1.5708	10.1623	...	0	0
2	9.01388	-2.33172	10.5	-1.5708	10.1623	...	0	3
3	9.01388	-2.33172	10.5	-1.5708	10.1623	...	0	3
⋮	⋮	⋮	⋮	⋮	⋮	...	⋮	⋮
1000	7.93404	0.904754	8.04407	1.64693	8.92419	...	-35.3793	1

The first tests showed that a pass is a frequent event, but owing to the short period of time of the actual passing, the total duration of the 'passing classes' is too short. In less than 10% of the time an actual pass is happening. But this is not sufficient to produce good results.

To get rules about the behavior of the opponent from such a rare event with the described decision tree method no error-based pruning can be done. Splitting the samples into the two not-passing classes dribbling and ball with the other team, leads to rules which are in more than 90% correct. With these values the error used to end the decision tree algorithm must be noticeably smaller than 10%.



```
(...)

(1)  if   series 8 > 1.000000
     and  series 0 < 1.530345
     and  series 0 < 0.862784
     and  series 2 < 3.559328
     then 0(1) 1(0) 2(0)

(2)  if   series 8 > 1.000000
     and  series 0 < 1.530345
     and  series 0 < 0.862784
     and  series 2 > 3.559328
     then 2(0.589744) 1(0.282051) 0(0.128205)

(3)  if   series 8 > 1.000000
     and  series 0 < 1.530345
     and  series 0 > 0.862784
     then 1(0.489796) 0(0.346939) 2(0.142857)

(4)  if   series 8 > 1.000000
     and  series 0 > 1.530345
     then 1(0.797297) 0(0.13964) 2(0.05630063)
```

Fig. 7. The passing behavior of the Brainstormers in a game against Robolog

However, if the according threshold is set to such a low value problems of overfitting occur. This means that the necessary generalization is lost and the tree exactly learns the samples given to the algorithm. But this is not what we want because the results should be used to adapt the own team to the opponent. Overfitted rules describe how the opponent has acted in special situations but not how his general behavior operates. This cannot be used to predict the future behavior of the opponent. Thus, another way to improve the results must be found.

The results of a learning algorithm can also be changed by modifying the input values. In this case the problem is obviously the qualitative abstraction. It does not assign enough pass classes. This is a result of the shortness of the pass event. So if there would be a possibility to increase the duration of such an event this should improve the results noticeably. A close look at the game reveals that the positions of the players does not change much between two cycles, thus the environment short before and after the pass is very similar to the one at the pass itself. Hence, they can also be assigned to the same pass class. As a result the pass event is not longer analyzed, but the situation leading to a pass. If the two cycles before and after the actual event are also assigned to the class, the decision tree can be built without the overfitting problem.

To test the method on passes again Robolog Koblenz, Karlsruhe Brainstormers and VirtualWerder were used. Again, ten games were played between Robolog and Brainstormers, VirtualWerder and Robolog and VirtualWerder and Brainstormers. The game Robolog against Brainstormers revealed for the pass behavior of the Brainstormers rules as in figure 7. The first rule showed that the Brainstormers had problems if they were attacked by two players (series $0 < 0.9$, series $2 < 3.6$). In this case they always lost the ball. But if the second opponent was more than 3.6m away they only lost the ball 12% of the time. The small value used to split series 0 shows that the Brainstormers react very late. The ball is already in the kickable area of the attacker.

In the ten games between Robolog and Brainstormers there were always similar values in the rules, but not all at all times and not in the same order. Although very similar rules to those above could be found in the half of all games, the values just differed slightly .

While analyzing the passing behavior of Robolog, the coach found rules like in figure 8. These rules reveal that Robolog tends to lose the ball if one of their own players is near to the ball carrier (series $4 < 0.5$m) except if the attacker is coming from behind (series $1 < -1.36$), in this case they are passing very successfully to a team mate. The problem of two Robolog player close to each other was revealed in nearly every game. It was found in the games against VirtualWerder as well.

```
if   series 8 < 1.000000
and  series 4 < 0.462778
and  series 1 < -1.356436
then 2(1) 0(0) 1(0)

if   series 8 < 1.000000
and  series 4 < 0.462778
and  series 1 > -1.356436
then 0(1) 1(0) 2(0)

if   series 8 < 1.000000
and  series 4 > 0.462778
then 1(0.613426) 0(0.208333) 2(0.157407)

(...)
```

Fig. 8. The passing behavior of Robolog in a game against Brainstormers

The tests with analyzing passes also showed that the difference in the quality of the teams influences the results. It is, however, not nearly as great as with the goalkeeper where it could happen that a goalkeeper did not have to move during a whole game which made an analysis of his behavior impossible. It is not as obvious with the passes because even with a weak opponent all teams still passed. Though VirtualWerder could not put much pressure on the Brainstormers there were only few interceptions and thus they have only seldom appeared in the results.

4.3 Related Work

Similar work has be done by Raines and colleagues (1999). They describe a program called ISAAC for off-line analysis that uses C5.0 and pattern matching to generate rules about the success of individual players and team cooperation in certain key events. Key events are events which directly effect the result of the game. The only one used in ISAAC is a shot at the goal. These events are analyzed, similar to the approach in this paper, with a decision tree algorithm. However, ISAAC has to be is used off-line, thus the program is not able to support real-time conditions. The team cooperation is analyzed by a pattern matching algorithm. This patterns are kicks of the ball by certain players which lead to a goal. The rules produced by ISAAC are intended support the development of the analyzed team. Therefore, they show how successful the team is in certain situations but not in which situations the players show which reaction.

An other off-line approach is described in [Wünstel et al., 2000], it uses self organizing maps to analyze the players movement and the players movement in relation to the ball. The trained map can be used to determine which kind of movement is done how often in a game. The results of this method show which kind of movements a player performs, but not in which situations he is doing so.

In the RoboCup 2000 the VirtualWerder coach [Visser et al., 2001] analyzed the opponent with an artificial neuronal network. The network was trained from log-files, from past competitions, to recognize 16 different team formations. To react on the opposing formation the coach was able to change the behavior of his own players.

Riley and Veloso (2002) use a set of pre-defined movement models and compare these with the actual movement of the players in set play situation. In new set play situations the coach uses the gathered information to predict the opponent agent's behavior and to generate a plan for his own players. The main difference to the approach described in this paper is that they analyze the movement of all players in set play situations, while the decision tree approach analyzes certain behaviors of single players and how successful they are in these situations.

5 Conclusion and future work

We showed that on-line learning is possible in time-critical environments as demonstrated in the simulation league. The idea is to see an object in the game as a time series. We applied a qualitative abstraction of those time series and used a new approach which is able to discreetize the time series in a way that the results are useable for symbolic learning algorithms. We implemented the approach and ran various tests in real games. We were using two scenarios to analyze the behavior of the goalkeeper and the pass behavior of opponent players. The discussion indicated that the knowledge derived from our approach is valuable and can be used for further instructions to players of the analyzed team. At present, results are generated every 1000 cycles, however, this depends on the situation to be analyzed.

At the moment we are developing our on-line coach in a way that he can use the results of the described approach and give instructions to his players. In order to use the collected rules and get advantage from them they still need to be processed in order to generate advice and to transmit it through the standard coach language to the players. For this purpose the rules should be refined. This process should e.g. reduce the number of attributes in a rule. One attribute should occur in one rule more than twice to mark a range of a value. Additionally, rules which cannot be transformed into advice for the own team can be deleted. The approach is very promising and we hope that first results can be seen on-line at the competitions in June.

If we try to generalize our approach the following statements can be made. Firstly, the proposed minimization heuristic has been successfully applied in a scenario for a qualitative substitution of sensors [Boronowsky, 2001]. It has been shown that qualitative cohesions between measurement-value time series are generally valid and that these cohesions can be discovered by automatic knowledge extraction procedures. The heuristic was also employed with respect to a qualitative analysis of a technical system. The scenario is purification of sewage water under experimental conditions dealing with two regulation circuits. They are regulating the pH value and the oxygen content. The results show that the regulation of both the pH value and the oxygen content can be modelled qualitatively.

Secondly, the method can be used to generate more rules skipping pruning techniques. Usually the amount of generated rules have to be decreased in order to derive comprehensible rules. This can done with the help of pruning methods while creating the decision tree. There are also ideas of how to automatically reduce the number of rules after being generated. This is investigated currently in a project and is subject of a masters thesis. However, sometimes pruning is not an option at all, e.g. if the number of generated rules are to small. In this case, we would be able to skip the pruning methods and therefore generate more rules. These rules should then enable a domain expert to either verify given hypothesis or producing hypothesis about a certain domain.

References

[Boronowsky, 2001] Boronowsky, M. (2001). *Diskretisierung reellwertiger Attribute mit gemischten kontinuierlichen Gleichverteilungen und ihre Anwendung bei der zeitreihenbasierten Entscheidungsbauminduktion.* PhD thesis, Department of Mathematics and Computer Science, University of Bremen, St. Augustin.

[Drücker et al., 2002] Drücker, C., Hübner, S., Visser, U., and Weland, H.-G. (2002). “as time goes by” - using time series based decision tree induction to analyze the behaviour of opponent players. In *RoboCup-01, Robot Soccer World Cup V*, Lecture Notes in Computer Science, Seattle, Washington. Springer-Verlag. in print.

[Fayyad and Irani, 1992] Fayyad, U. and Irani, K. (1992). On the handling of continuous-valued attributes in decision tree generation. In *Machine Learning*, volume 8, pages 87–102.

[Murray, 1999] Murray, J. (1999). My goal is my castle – die höheren fähigkeiten eines robocup-agenten am beispiel des torwarts.
http://www.uni-koblenz.de/ag-ki/ROBOCUP/PAPER/papers.html.

[Quinlan, 1993] Quinlan, J. (1993). *C4.5 Programs for Machine Learning*. Morgan Kaufmann.

[Raines et al., 1999] Raines, T., Tambe, M., and Marsella, S. (1999). Automated assistants to aid humans in understanding team behabiors. In Veloso, M., Pagello, E., and Kitano, H., editors, *Proceedings of the Third International Workshop on Robocup 2000, Robot Soccer World Cup IV*, volume 1856 of *Lecture Notes in Computer Science*, pages 85–102, Stockholm, Sweden. Springer-Verlag.

[Riley and Veloso, 2002] Riley, P. and Veloso, M. (2002). Recognizing probabilistic opponent movement models. In *RoboCup-01, Robot Soccer World Cup V*, Lecture Notes in Computer Science, Seattle, Washington. Springer-Verlag. in print.

[Visser et al., 2001] Visser, U., Drcker, C., Hbner, S., Schmidt, E., and Weland, H.-G. (2001). Recognizing formations in opponent teams. In Stone, P., Balch, T., and Kraetschmar, G., editors, *RoboCup 2000, Robot Soccer World Cup IV*, volume 2019 of *Lecture Notes in Computer Science*, pages 391 – 396, Melbourne, Australia. Springer-Verlag.

[Wünstel et al., 2000] Wünstel, M., Polani, D., Uthmann, T., and Perl, J. (2000). Behavior classification with self-organizing maps. In Stone, P., Balch, T., and Kraetschmar, G., editors, *RoboCup 2000, Robot Soccer World Cup IV*, volume 2019 of *Lecture Notes in Computer Science*, pages 108–118, Melbourne, Australia. Springer-Verlag.