# Learning Situation Dependent Success Rates Of Actions In A RoboCup Scenario

Sebastian Buck

Martin Riedmiller

Technical University of Munich Computer Science Department IX Institute for Logic, Complexity Orleansstr. 34 D - 81667 München, FRG buck@in.tum.de

University of Karlsruhe and Deductive Systems D - 76128 Karlsruhe, FRG riedml@ira.uka.de

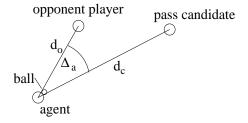
**Abstract.** A quickly changing, not predictable environment complicates autonomous decision making in a system of mobile robots. To simplify action selection we suggest a suitable reduction of decision space by restricting the number of executable actions the agent can choose from. We use supervised neural learning to automaticly learn success rates of actions to facilitate decision making. To determine probabilities of success each agent relies on its sensory data. We show that using our approach it is possible to compute probabilities of success close to the real success rates of actions and further we give a few results of games of a RoboCup simulation team based on this approach.

### 1 Introduction

The RoboCup soccer server [Noda98] offers a couple of low level commands for soccer agents to choose from each 100 ms. Mainly they have the following options: turn (angle), dash (power), kick (power) (angle). Only a player possessing the ball can kick. Our focus lies on more complex high level actions such as pass, shoot2goal or go2ball which represent a sequence of low level commands each. [Matsubara96] investigated the learning of success rates of pass play and goal shots in soccer server in a certain game situation using one feed forward network with one output unit for each action. We compute an explicit situation dependent success rate for n defined actions using n neural networks (one for each action). From all promising actions (estimated success rate exceeds threshold) the one ranked highest in a priority list is chosen to be executed. In the following we substantiate our architecture and present our learning algorithm and learning environment. Last we show empirical results and results of games and complete with a final discussion.

### 2 Architecture Model

If we treat the task of playing soccer as an optimization problem the aim is to control our agents in the given environment such that they score more goals



**Fig. 1.** The features of the network computing the success rate of pass play: distance to the pass candidate  $d_c$ , angle of an opponent player to the line to the pass candidate  $\Delta_a$  and the distance of this opponent player  $d_o$ .

than the opponent team does. We can estimate the number of possible policies by discretising the angle and the power value of the low level commands: Assuming 72 possible angles (5 degree steps) to turn to or to kick to and 10 power levels to dash with or to kick with we get 72+10+720 different commands to choose from for a player possessing the ball at one time step. This means we have up to  $802^{3000}$  different policies over a period of five minutes for only **one** agent. This forces us to reduce the number of possible choices per time step. To do this we introduce a number of actions  $A = \{a_1, ..., a_n\}$  from which the agent can choose. Similar architectures have also been used in [Dorer99] where actions are represented as consummatory nodes in a Behavior Network, in [Stone99] and in earlier works such as [Stone 98] and [Burkhard 97] as well. Defining the actions dribble (with parameters left, right, forward), shoot2goal (with parameters left/right corner, center of goal) and pass (with teammate positions as parameters) for a player possessing the ball we already cover a wide space of useful possibilities having left only about a dozen options instead of 802. The defined actions are ranked in a priority list A according to their gain  $G(a_i)$ : A goal shot is better than a pass, a pass to a player near the opponent goal is better than a pass back and so on. For each action  $a_i \in A$  a success rate  $P(a_i)$  is computed. From all promising actions  $(P(a_i))$  exceeds threshold  $\Theta_{a_i}$  the one  $a_{ex}$  with the highest position in the priority list is executed:

$$P(a_{ex}) \ge \Theta_{a_{ex}} \land \not\exists a_j \in A : (P(a_j) \ge \Theta_{a_j} \land G(a_j) > G(a_{ex})) \tag{1}$$

The thresholds  $\Theta_{a_i}$  are initially tuned manual but can be adapted online depending on the opponents strength. Obviously we need a default action in case of low success rates of all other actions  $(P(a_i) < \Theta_{a_i} \forall i)$ . Regarding a player holding the ball this could be dribble (forward) for example. The next section explains how to compute the success rates  $P(a_i)$ .

# 3 Learning Success Rates

Within our first experiments we determined the  $P(a_i)$  by simple manually defined functions based on attributes like  $goal\_is\_near$  or  $teammate\_is\_free$  which

did surprising well. But to reach values of  $P(a_i)$  close to the real success rates neural learning was chosen. Therefore we apply multi layer perceptrons as described in [Hertz91] (one for each action) with one or two hidden layers and one output unit representing the success rate. Moreover we replace the backpropagation algorithm with Rprop [Riedml93] because of its more sophisticated step size computation in gradient descent which is done on the quadratic error function

$$E(T) = \frac{1}{2} \sum_{t \in T} (net_{a_i}(X_t) - Y_t)^2$$
 (2)

where T is the training set and  $t = (X_t, Y_t)$  is a single pattern.  $net_{a_i}$  denotes the neural network corresponding to action  $a_i$ . The inputs  $X_t$  of those networks are action specific features. In figure 1 the features of pass play are given as an exemplar. In this case a pattern t consists of  $((d_c, \Delta_a, d_o), Y_t)$ . The pass network computes success rates for all known opponent players and the lowest network output is returned as final estimation of the success rate of pass play:

$$P(pass) = min_{opp} \{ net_{pass}(X_{opp}) \}$$
(3)

where  $X_{opp}$  are the features of a pass to a certain candidate considering the opponent player opp. Analogously but with different features estimations of success rates of goal shots are determined.

For instance to create training patterns  $(X_t, Y_t)$  of pass play we automatically generate random situations with one player (TeamA) playing a pass, one opponent player (TeamB) and one pass candidate (TeamA) both trying to intercept the ball. If the ball reaches the candidate the features  $X_t$  are saved with  $Y_t = 1$  else with  $Y_t = 0$ . Features  $X_t$  are measured at the moment the agent starts to play a pass.

In practice we learned success rates of the actions pass, shoot2goal and go2ball. Therefore networks with 3-5-1, 3-5-1 and 5-8-5-1 topology lead to the best results. The sizes of the training sets ranged between 800 (3-5-1 networks) and 8000 (5-8-5-1 network). In all cases less than 100 learning epochs were enough to reach a minimum error on the test sets (200 and 2000 patterns).

Contrary to pass and shoot2goal success rates of go2ball were only learned without opponent players in order to designate one player of the own team to go to the ball. Even if an opponent agent holds the ball it seems reasonable that at least one player attacks him. For that reason teammates instead of opponent players are included in computation of P(go2ball).

# 4 Results

In order to evaluate our concept we played simulation games against different teams such as [Burkhard97], [Stone98] and our own team in which we measured the real success rates  $S(a_i)$  of actions  $a_i$  resulting from different adjustments of the threshold parameters  $\Theta_{a_i}$ . Actions were only executed if  $P(a_i) \geq \Theta_{a_i}$ .  $S(a_i)/\Theta_{a_i}$  ratios of the actions pass, shoot2goal and go2ball are plotted in figures

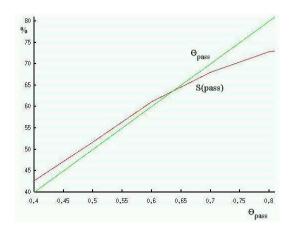


Fig. 2. Success rates of pass play in games: The real success rate S(pass) depending on  $\Theta_{pass}$  was measured. One can see that in a certain area S(pass) and  $\Theta_{pass}$  are nearly identical.

2, 3 and 4. Pass play was tested with  $\Theta_{a_i} \geq 40\%$  only because in practice we need high success rates in pass play. Likewise goal shots were tested with  $\Theta_{a_i} \leq 40\%$  because 40% statistically means already 0.4 goals.  $S(a_i)/\Theta_{a_i}$  is about 1 for pass and shoot2goal but only about  $\frac{1}{2}$  for go2ball because in training success was only learned against teammates. In real games opponent players try to get the ball as well but our networks used in this experiments do **not** depend on opponent players because it makes sense to go towards the ball even if an opponent agent holds it. All results are based on 1000 executed actions per adjustment of  $\Theta_{a_i}$  ( $S(a_i)$  is interpolated). Real time problems caused by too large or too many networks never occurred even with five agents on one 166MHz AMD linux machine. Main temporal effort of this algorithm appears while learning but not in application.

In addition to our statistics our concept was quite successful in games of our team *Karlsruhe Brainstormers* against some simulator league teams of 1999:

Location	Opponent team	Ranking at	
		RoboCup 99	
1999 Autumn Japan Camp	11 Monkeys [Yamamoto99]	4.	1:0
1999 Autumn Japan Camp	Essex Wizards [Kostiadis99]	3.	3:1
RoboCup 1999	Essex Wizards [Kostiadis99]	3.	0:1
1999 Autumn Japan Camp	Magma Freiburg [Dorer99]	2.	1:3

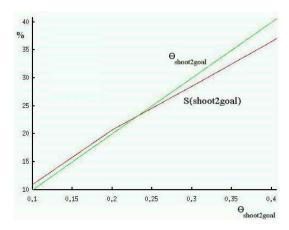
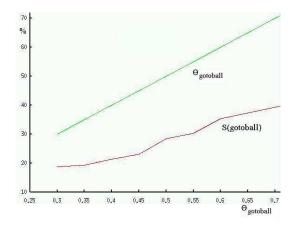


Fig. 3. Success rates of goal shots in games: The effective success rate S(shoot2goal) depending on  $\Theta_{shoot2goal}$  is close to  $\Theta_{shoot2goal}$  over the full observed period.



**Fig. 4.** Success rates of go2ball in games: S(go2ball) is about half  $\Theta_{go2ball}$ . That makes sense since in the training environment success was learned only versus teammates but not versus opponent players.

Furthermore Karlsruhe Brainstormers made with 32:0 in RoboCup 1999 the highest match win ever reached in a RoboCup tournament.

## 5 Conclusion

Decision making in a system of autonomous mobile robots usually leads to a huge number of possible policies. We can simplify the problem by pruning the number of possible choices from which a robot has to select. Indeed we should not ignore that through this we may lose some good or even optimal policies. Thus we have to select the possible actions carefully. In our case it is imaginable which actions a robot needs at least to play soccer.

The learning of success rates seems to be a qualified way to choose from a couple of possible actions. In most cases results prove that it is feasible to obtain values which resemble the measured success rates. Of course those results of  $S(a_i)$ cannot be exact  $\Theta_{a_i}$  because of the noisy environment and the limited view and information of a soccer agent. Maybe results become better with a superior perception or a different set of features. The specific features are chosen manually so far. The most important parameters of this approach are the thresholds  $\Theta_{a_i}$ which remain to be tuned by human. These parameters are not some magic numbers but a set of values one can imagine what they are doing: If  $\Theta(pass)$ is set to 0.7 pass play is only executed if the estimated success rate of pass play P(pass) is at least 70%. It's understood that this approach means only a one step optimization because future actions are not regarded so far. This could be changed by defining the gain  $G(a_i)$  (which is responsible for the ranking of the priority list) dependent on the success rates of future actions. However it is difficult to determine the success rates of future actions because of the quickly changing environment. Therefore it would be necessary to optimize for all possible future situations. Henceforth Karlsruhe Brainstormers will follow this idea. In parallel some of this work will be included in mid size league robots Agilo Robo Cuppers Munich [Klupsch99]. It will be a challenge to automatize the generation of training patterns with real robots.

### References

- [Burkhard97] Burkhard et al. (1997): AT Humboldt Development, Practice and Theory, in H. Kitano, editor, RoboCup-97: Robot Soccer World Cup I, Springer
- [Dorer99] Dorer (1999): Behavior Networks for Continuous Domains using Situation-Dependent Motivations, Proceedings of the IJCAI, Stockholm
- [Hertz91] Hertz, Krogh, Palmer (1991): Introduction to the theory of neural computation, Addison Wesley
- [Klupsch99] Klupsch et al. (1999): Agilo Robo Cuppers: Robo Cup Team Description, in Coradeschi et. al., editors, Robo Cup-99: Team Descriptions
- [Kostiadis99] Kostiadis and Hu (1999): Essex Wizards, in Coradeschi et. al., editors, RoboCup-99: Team Descriptions

- [Matsubara96] Matsubara, Noda, Hiraki (1996): Learning of Cooperative actions in multi-agent systems: a case study of pass play in Soccer, AAAI-96 Spring Symposium (Adaptation, Coevolution and Learning in Multiagent Systems)
- [Noda98] Noda et al.: Soccer Server Manual Ver 4 Rev. 00, November, 1st, 1998, http://ci.etl.go.jp/~noda/soccer/server/index.html
- [Riedml93] Riedmiller and Braun (1993): A direct adaptive method for faster back-propagation learning: the Rprop algorithm, Proceedings of the ICNN, San Francisco
- [Stone et. al. (1999): The CMUnited-99 Simulator Team, in Coradeschi et. al., editors, RoboCup-99: Team Descriptions
- [Stone98] Stone and Veloso (1998): Team-partitioned, opaque-transition reinforcement learning, in M. Asada and H. Kitano, editors, RoboCup-98: Robot Soccer World Cup II, Springer
- [Yamamoto99] Yamamoto and Kinoshita (1999): Team 11monkeys Description, in Coradeschi et. al., editors, RoboCup-99: Team Descriptions