A Neural Coach for Teaching Robots Using Diagrams

Bastian Hecht, Mark Simon, Oliver Tenchio, Fabian Wiesel, Alexander Gloye, Raúl Rojas
Freie Universität Berlin
Institut für Informatik
Takustraße 9, 14195 Berlin
Germany
{bhecht,simon,tenchio,gloye,rojas}@inf.fu-berlin.de

Abstract

In this paper we show how to train soccer robots using static game situations in diagrams arranged by a human coach. Rather than programming every detail by hand, we let the robots learn from strategic examples sketched by the coach. With our approach, the coach defines new game positions and indicates to the players how to react to them, like in real soccer. We have implemented a management tool to collect and organize all the game positions entered by the coach. The game situation is encoded as a feature vector, which is used to train a neural network. The network learn to generalize and give advice on the best option for a player. The general method is illustrated with the specific case of robots learning to pass. The method can be generalized to other tasks and to several networks encoding different game strategies.

1 Motivation

RoboCup robots are usually programmed by hand. Learning techniques, such as reinforcement learning, have been used for several years in the simulation league [3]. In the robotic leagues it is more difficult to automatically learn high-level skills, and therefore learning has been mostly used to allow robots to automatically adapt the parameters of low-level skills [2]. It is also clear why: we cannot let real robots play against themselves hundreds of times, so that they learn to behave successfully.

An alternative could be a simulation, but an exact model of the robots is never so exact that a simulation could be used as a complete substitute [9]. On the one hand, it is hard to have an errorless prediction of the driving behavior of real robots. On the other hand, very small changes in hardware can lead to significantly different characteristics, such as more or less ball control when driving.

In this paper we investigate a second option. We want to supply our small-size robots with our own human knowledge about soccer. Until now, we have achieved respectable robot behavior mostly using manual hard coding and tuning the code in long and difficult "training" sessions. We would like to teach the robots in the same way a human coach explains plays to human players: using static diagrams of what constitutes a good and what constitutes a bad move. What we propose is that a human coach draws interesting game situations, for example for passing [4], and then assigns them a "good" or "bad" grade. The computer should then learn to generalize from such examples to new and unseen game situations.

Entering enough examples into the system, it is then possible to train a neural network which can achieve the desired generalization. The coach trains the robots with examples, and the robots learn to do the right thing. Moreover, by keeping separate databases of offensive or defensive strategies, it is possible to train several neural networks for different styles of play. We then can integrate an external agent (a coach, as in the simulation league), which can provide advice on the best strategy for the current adversary [2]. The robots can then switch their strategy dynamically according to this advice.

2 Setting up Training Examples of Game Situations

The first step for training the robots with our approach is to set up some examples of possible game situations. For this we can use our simulator for the small-size robots. From now on, let us assume that we want robots to learn how to pass the ball (and receive it). The player with the ball will be called the "passer", and the player receiving the ball will be called the "receiver". Figure 1 shows a scene in which player 0 should pass the ball to player 1, which is waiting for the pass. With our simulator, the user can set up such a game situation by dragging players from each team to the desired position.

In what follows, we focus only on a passer and a single available receiver. Later on, we generalize our techniques to take all other field players into account.

Once an example has been entered by the coach, we save all relevant information in an XML-file. In the experiments described in this paper, we have used static situations, where the speeds of all robots and the ball are zero. However, the same general approach can be applied to dynamic situations. For every stored game situation, we associate with it information that reflects our belief on the correct decision for the the potential passer (giving the pass or not). In the case of the passing game, we store a real number $x \in [-1; 1]$. In our system, we apply the convention that when $x \leq 0.0$ dribbling is desirable, and that when $x > 0.0$ passing is desirable.

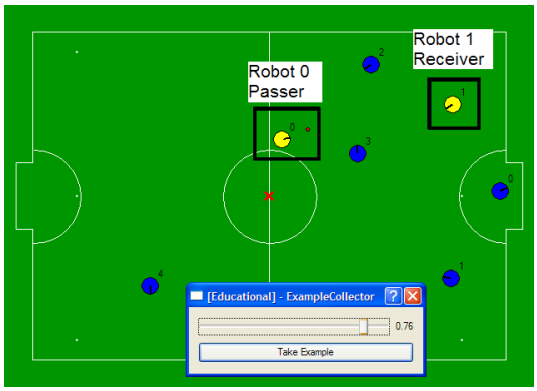

Figure 1: A static game situation entered by the human coach

3 Encoding Relevant Features

In order to generalize from the stored examples, it is crucial to look not at the coordinates of the robots, but to some features of the game situation. If we would just encode the coordinates of the robots on the field, and would give this information to a neural network, the net would have extreme difficulties trying to generalize to new game situations. A barrier of players, for example, is a feature that looks similar in almost all places in the field, although the coordinates are of the robots can be very different.

We need to encode the field using a numerical feature vector. For example, a possible feature is the free space around the ball receiver. This feature is very relevant because it does not pay to give a pass to a player which will be trapped.

By encoding the game situation with a feature vector, we necessarily lose information because we cannot reconstruct all robots' positions from the features, but we obtain a more abstract view of the field. The information given to the net has a better format, since important field aspects are encoded numerically.

Back to our passing example. Conceptually, the features we use are split up into two parts, the features having to do with dribbling (that is, driving with the ball), and the features having to do with ball reception. The features are independent from the training method used.
The features associated with dribbling are:

1. Dribbling freedom

   This parameter describes the space available for dribbling with the ball, before an opponent appears (in the direction of the opponent's goal). This parameter reflects the time that it would take the nearest opponent to interfere with the dribbling path of my robot.

2. Dribbling angle

   This is the angle defined by the position of the passer, the middle of the opponent's goal line, and the nearest corner of the field. A robot positioned at an opponent's corner, for example, has a dribbling angle of zero. A robot in the middle of the field, has a dribbling angle of 90 degrees. A larger dribbling angle means that more space for dribbling is available.

3. Dribbling distance to the goal

   This is the just distance from the passer to the goal line.

The next five features describe the reward obtained from passing. We can visualize these features in the following way: The player on the left has the ball, and ponders whether to give a pass. The player on the right is only a symbol for a receiver at one position. In fact, we displace the pass receiver over a grid of $17x21$ points covering the field, and calculate at each vertex the features. The results are illustrated in the following figures.

4. Space available

   This parameter encodes the distance from the receiver up to the nearest opponent. It is large when the opponents are far away, it is small when an opponent is near. Fig. 2 shows, with a white marker, those portions of the field where there is much space available, and with black the space dominated by opponents.

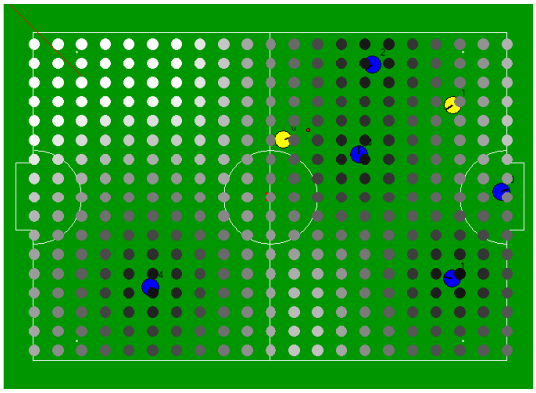

Figure 2: Space available: White areas of the field are relatively opponent-free, dark areas are not

5. Passing angle

   This is the same feature we had above (dribbling angle), but now for the receiver. A high passing angle means that it is good to receive passes, for example in the middle of the field. A low passing angle means that it is not so good to receive a pass, for example, at the corners or near the sides of the field. Fig. 3 shows with a white marker those positions in the field which offer a good passing angle, and in dark those positions with a worse passing angle.

6. Minimal tangent distance

   This value is the shortest distance an opponent has to move in order to block a pass. Fig. 4 shows, in black, those field areas where a pass can go through. The white areas can be blocked by the opponents. In the example, the receiving robot (lower right) is too near to an opponent. It would be better if the receiving robot was located at some point in the diagonal corridor going from the middle of the field to the upper right.

7. Waiting time during passing

   This feature is the time elapsed after an opponent has moved to a new position, where a pass can be blocked,

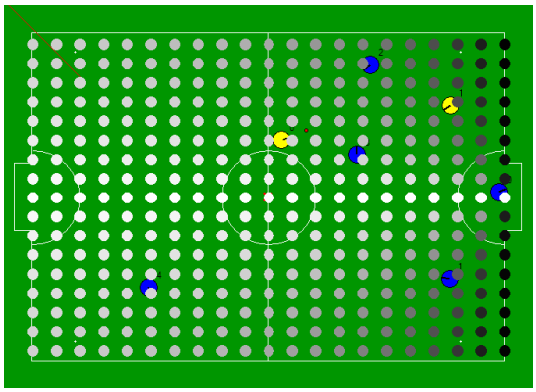

Figure 3: Passing angle: White areas of the field provide a good passing angle, dark areas do not

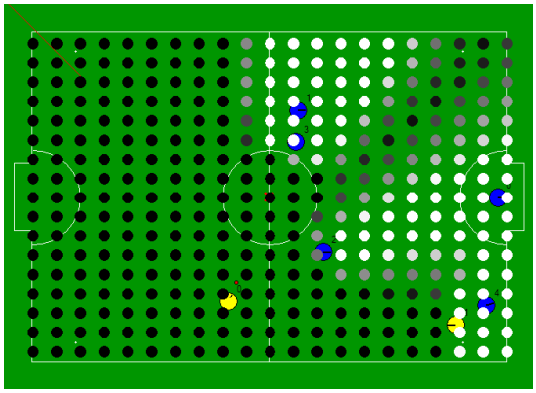

Figure 4: Minimal tangent distance: In dark areas of the field the distance to block a pass is large, in white areas not

and the instant where the ball arrives. An opponent with a large waiting time can block a pass easily. An opponent with negative waiting time cannot reach the ball. Fig. 5 shows an example where the dark areas cannot be easily reached by the opponents to stop a pass, while the white or gray areas can be reached easily. This is probably one of the main criterions which need to be used for deciding to give a pass or not.

8. Dribbling freedom for the receiver after a pass

   This is the same parameter as "dribbling freedom" for the robot with the ball, but now for the robot receiving the pass. Fig. 6 shows the regions of the field where receiving a pass provides high reward (in black) and low reward (in white).

It is worth noting that all these features encode symmetrical field positions with the same numbers. If we had stored the coordinates of the robots, we would have to store all the symmetrical cases every time we enter an example. The features described above take care of encoding all symmetrical field positions with the same feature vector. The classifier has an easier task, once the symmetries of the learning problem have been incorporated in the encoding.

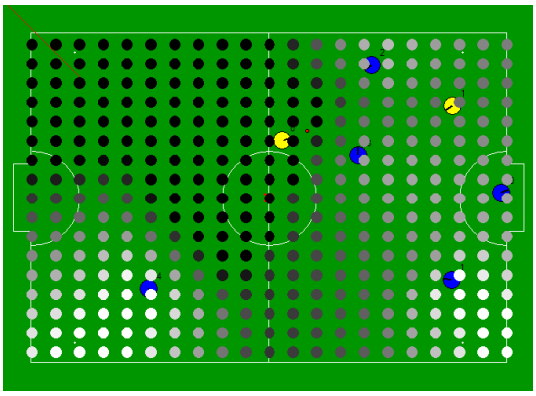

Figure 5: Waiting time: Opponent robots in white areas can wait longer for the ball when a pass is coming

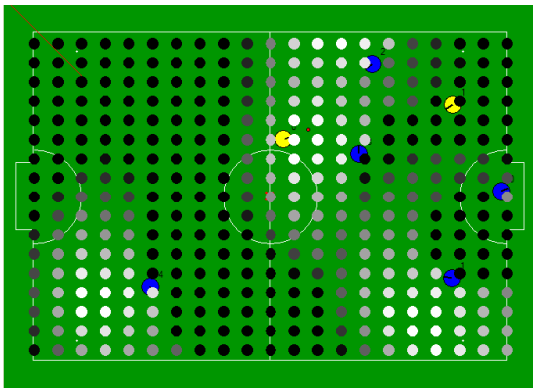

Figure 6: Dribbling freedom for the receiver. Dark areas indicate much free space, bright areas are covered by opponents

4 Training neural networks

We have written a tool to keep track of all the examples of game situations defined by a human coach (or several human coaches). We can group the examples in several categories to have a better overview of them and to activate and deactivate particular example groups.

By activating and deactivating groups, we can train different neural networks to encode different behaviors. For example, if an opponent has the ability to block long passes across the field, we can deselect all examples where long passes are considered "good", and we can train a network which behaves essentially as the original one, except for its reluctance now to propose long passes across the field.

We achieved good classification results using a three-layer feed forward network [6]. The network has 8 input nodes - the features seen in section 3 - and a hidden layer, whose dimension can be set arbitrarily. Right now, we usually have 14 hidden nodes. The output node emits a value that indicates the action that should be taken. An output in the interval $(-\infty; 0]$ is interpreted as "Do not pass", while an output in $(0; \infty)$ as "Do pass". The neural network we use, has already reached a size where it is questionable whether it would be good to let it grow further. One would need too many examples to

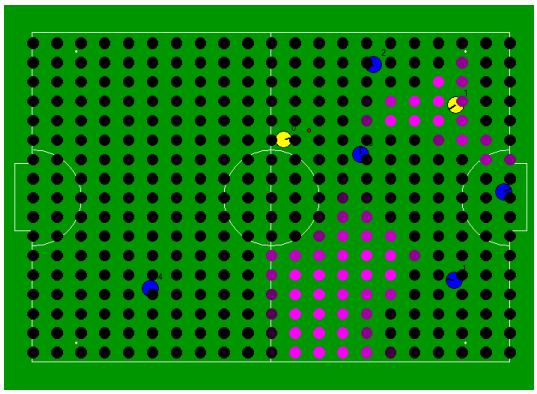

Figure 7: This is the output of a trained neural coach. If a teammate stands in one of the 2 pink areas, the left player with the ball should pass.

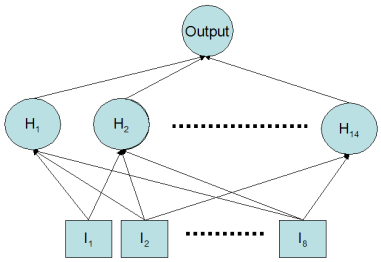

Figure 8: The topology of the neural network used as passing coach

cover all degrees of freedom of the network. Right know, we have stored around 100 examples in our system. The effort is worth it: game situations are created quickly with our simulator, and the results we obtain have good quality.

The trained networks can be saved, and it is possible to use them for behavior control. In our current system, the eventual passer first ponders shooting a goal. This behavior inhibits all others. If shooting is unfeasible or not so good, the robot considers whether to dribble or to pass. The system iterates over all team mates (as possible receivers of a pass), and asks the coach whether it would be good to pass or not. If there are more than one well positioned receivers, the passer robot selects the one for which turning towards it is easier.

5 Conclusion

We have developed a “neural coach” for our small-size robots. The coach is a neural network which accepts game situations encoded as a feature vector, and which provides as output a number reflecting the best alternative: dribbling with the ball or passing. The robot with the ball can periodically ask the neural network which is the best strategy, and can apply it.

Our behavior control system has grown with the years and contains many parameters which must be tuned by hand. It is difficult to modify them when the hardware changes, also because the many programmers work with our system. Our coaching tool is a decisive step towards abandoning such

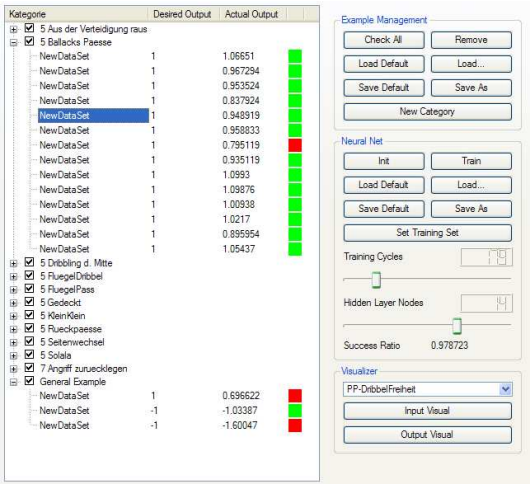

Figure 9: Our management tool saves static examples of game situations. The examples can be deleted, or organized in groups which correspond to alternative offensive strategies.

hand-tuned implementations, in favor of a more general approach.

Many other game decisions could be modelled in the way described in this paper, like for example the team formation, or individual behaviors of a robot (”I am the goalkeeper and see a opponent dribbling to my goal. My defenders are far away. Shall I come out of my goal to decrease the shooting angle or not?”).If the behavior control system can also learn the low-level skills, such as driving, or dribbling with the ball, using reinforcement learning or other machine learning algorithms, one obtains a more versatile robotic platform, and code which is easier to manage and maintain.

References

[1] [Fidelman and Stone, 2004] Peggy Fidelman and Peter Stone. ”Learning Ball Acquisition on a Physical Robot”. In 2004 International Symposium on Robotics and Automation (ISRA), August 2004.

[2] [Kuhlmann and Stone, 2005]. Gregory Kuhlmann and Peter Stone. ”The UT Austin Villa 2003 Champion Simulator Coach: A Machine Learning Approach”. In Daniele Nardi, Martin Riedmiller, and Claude Sammut, editors, RoboCup-2004: Robot Soccer World Cup VIII, Springer Verlag, Berlin, 2005. To appear.

[3] [Lauer and Riedmiller, 2004] M. Lauer and M. Riedmiller. ”Reinforcement Learning for Stochastic Cooperative Multi-Agent Systems”. In Proc. of AAMAS 2004, New York, USA

[4] [Kok et al., 2003] Kok, Jelle R.; Spaan, Matthijs T. J.; Vlassis, Nikos. ”Multi-robot decision making using coordination graphs”. In Proceedings of the 11th International Conference on Advanced Robotics, ICAR’03, pp. 1124-1129, Coimbra, Portugal, June 2003.

[5] [Lipson and Shpitalni] Lipson, H.; Shpitalni, M. ”Conceptual Design and Analysis by Sketching”. In Journal of

AI in Design and Manufacturing, Vol. 14, pp. 391-401, 2000

[6] [Raul Rojasm, 1996] Rojas, Raul. ”Neural Networks - A Systematic Introduction”. Springer-Verlag, Berlin, 1996.

[7] [Egorova et al., 2004] Egorova, Anna; Gloye, Alexander; Göktekin, Cüneyt; Liers, Achim; Luft, Marian; Rojas, Raúl; Simon, Mark; Tenchio, Oliver; Wiesel, Fabian. “FU Fighters Small Size Team 2004”. in N.N. (editors): RoboCup-2004: Robot Soccer World Cup VIII, Springer, 2005.

[8] [Behnke et al., 2000] Behnke, Sven and Rojas, Raúl. “A Hierarchy of Reactive Behaviors handles Complexity”. in: Proceedings of: Balancing Reactivity and Social Deliberation in Multi-Agent Systems, a Workshop at ECAI 2000, the 14th European Conference on Artificial Intelligence, Berlin, 2000.

[9] [Gloye et al., 2005] Gloye, Alexander; Göktekin, Cüneyt; Egorova, Anna; Tenchio, Oliver; and Rojas, Raúl. “Learning to Drive and Simulate Autonomous Mobile Robots”. in N.N. (editors): RoboCup-2004: Robot Soccer World Cup VIII, Springer, 2005.