Broad Learning from Narrow Training: A Case Study in Robotic Soccer

Peter Stone and Manuela Veloso

November 8, 1995
CMU-CS-95-207

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

The range of unseen instances that can be successfully classified by a learning algorithm is determined not only by the distribution of the training data, but also by the parameters of the function to be learned. With the right parameters, learning in a certain region of the state space can generalize to completely different areas with no retraining. We demonstrate the power of using well-chosen inputs to (and outputs from) neural networks by conducting experiments in our robotic soccer domain. We train an agent to shoot a moving ball into a goal in a specific situation and end up with a general shooting behavior that is much more widely applicable.

This research is sponsored by the Wright Laboratory, Aeronautical Systems Center, Air Force Materiel Command, USAF, and the Advanced Research Projects Agency (ARPA) under grant number F33615-93-1-1330. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Wright Laboratory or the U. S. Government.

Keywords: Robotic Soccer, Neural Networks, Flexible Inputs

1 Introduction

The range of unseen instances that can be successfully classified by a learning algorithm is determined not only by the distribution of the training data, but also by the parameters of the function to be learned. That is to say, any learning algorithm is representation-dependent: with inappropriate representations, learning algorithms will not reach their full potential, but with well-chosen representations they can be quite powerful.

In particular, Neural Networks (NNs) have been successfully used to handle problems in motion control from pole balancing [12] to driving autonomous vehicles [7]. After training, they can generalize to previously unseen situations. However, the true extent of their flexibility is limited by the input representation. If their input is specific to a given situation in any way, then they will not generalize beyond that situation. For example, Pomerleau describes a situation in which his autonomous vehicle can learn to turn left, but then cannot turn right [7]. Therefore it is the job of the user to provide, if possible, inputs to a NN (or other learning algorithm) that are maximally flexible.

In this paper we illustrate and demonstrate the power of flexible inputs for the task of learning to shoot a moving ball into the goal in a robotic soccer domain: one team member passes the ball in front of another, which in turn must time its acceleration and adjust its heading so that it redirects the ball into the goal. This learning task is an example of a low-level *behavior* in robotic soccer: it is a complex individual task that is prerequisite for higher-level team-oriented strategy issues. We train a player to score in a narrow situation and then test how broadly applicable the learned behavior is. We use a NN to train the shooting teammate to accomplish this task, but we do so in such a way that the learning applies to a much broader range of situations than are covered by the initial training. The secret of this flexibility is our choice of inputs to the NN.

This paper is organized as follows. Section 2 situates this work within the growing body of robotic soccer research and briefly describes our simulator. Section 3 describes our experimental setup. Sections 4–7 contain details of the experiments along with results. Section 8 provides some discussion and conclusions. The contributions of this paper include creating a flexible representation for learning behaviors in a real-time control domain, such as robotic soccer; learning a particular complex behavior with a robust NN; and giving extensive empirical validation supporting the effectiveness of the representation used.

2 Robotic Soccer

Robotic soccer has recently been emerging as a challenging topic for AI researchers interested in machine learning (ML), multi-agent systems, reactive behavior, strategy acquisition, and several other areas of AI. This section briefly summarizes previous research and describes our research platform.

2.1 Related Work

Sahota's Dynamo system [11] pits two remote control cars against one another in a game of 1 vs. 1 soccer. All of the behaviors are hard-wired, as is the decision mechanism used to choose

among the behaviors. Although this approach worked well for a specific task, ML is needed in order to avoid the cumbersome task of hard-wiring the robots for every new situation.

Also working within the Dynamo project, Ford et. al. [4] began learning the decision mechanism for these same hard-wired behaviors. They used a Reinforcement Learning (RL) approach, with *sensory predicates* used to reduce the size of the state space in a natural way. Our predicates used as inputs to NNs are somewhat analogous to their use of predicates for RL. So far all of the work in the Dynamo project has used Sahota's hard-wired behaviors.

On the other hand, Asada has used RL to learn a shooting behavior [1]. His framework differs from Dynamo in that it uses on-board sensors as opposed to an overhead camera. Again by reducing the state space significantly, he was able to use RL to learn to shoot a stationary ball into a goal. His best result in simulation is a 70% scoring rate. He has also done some work on combining different learned behaviors with a separate learned decision mechanism on top [2].

Our work builds on previous work by learning a more difficult behavior: shooting a *moving* ball into a goal. Our long-term goal is to create a team of agents with entirely learned behaviors and decision mechanisms that will be able to compete in the planned RoboCup event at IJCAI '97 [5]. This event will have both a real-world and simulated event in which researchers from around the world will be able to compare systems and methods.

2.2 The Simulator

Although we plan to participate in both events, the real system of model cars remotely controlled from off-board computers is still under development. Our research is currently conducted in a simulator of the physical system.

Both the simulator and the real-world system are based closely on systems designed by the Laboratory for Computational Intelligence at the University of British Columbia [3]. In particular, the simulator's code is adapted from their own code, for which we thank Michael Sahota whose work [11] and personal correspondence [10] has been motivating and invaluable. The simulator facilitates the control of any number of agents and a ball within a designated playing area. Care has been taken to ensure that the simulator models real-world responses (friction, conservation of momentum, etc.) as closely as possible. Sensor noise of variable magnitude can be included to model more or less precise real systems. A graphic display allows the researcher to watch the action in progress, or the graphics can be toggled off to speed up the rate of the experiments. Figure 1(a) shows the simulator graphics.

The simulator is based on a client-server model in which the server models the real world and reports the state of the world while the clients control the individual agents (see Figure 1(b)). Since the moving objects in the world (i.e. the agents and the ball) all have a position and orientation, the simulator describes the current state of the world to the clients by reporting the x, y, and θ coordinates of each of the objects indicating their positions and orientations. The clients periodically send throttle and steering commands to the simulator indicating how the agents should move. The simulator's job is to correctly model the motion of the agents based on these commands as well as the motion of the ball based on its collisions with agents and walls. The parameters modeled and the underlying physics are described in more detail in Appendix A.

The clients are equipped with a path planning capability that allows them to follow the shortest path between two positions [6, 9]. For the purposes of this paper, the only path planning

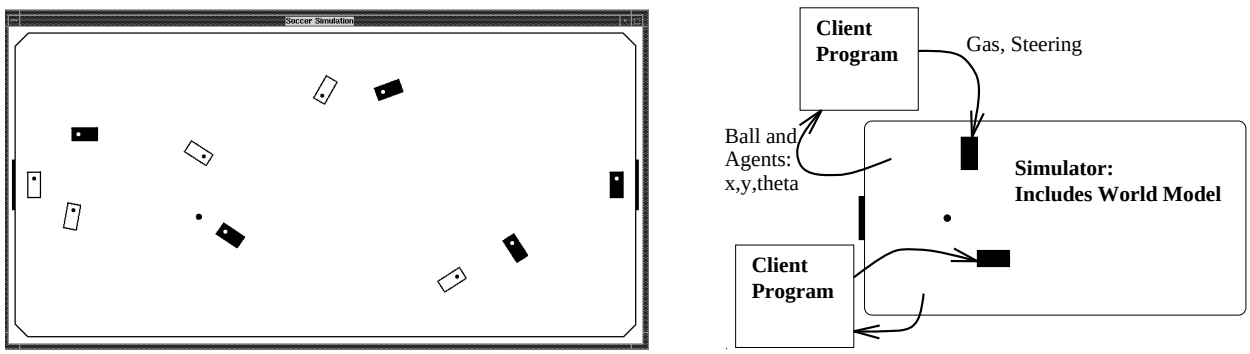


Figure 1: **(a)** The graphic view of our simulator. Eventually teams of five or more agents will compete in a real-time game of robotic soccer. **(b)** The interface between the clients and the simulator.

needed is the ability to steer along a straight line. This task is not trivial since the client can only control the agent's motion at discrete time intervals. Our algorithm controls the agent's steering based on its offset from the correct heading as well as its distance from the line to be followed. This algorithm allows the agent to be steering in exactly the right direction within a small distance from the line after a short adjustment period. Thus the agent is able to reliably strike the ball in a given direction. For more details on the line following method, see Appendix B.

3 Learning to Shoot a Moving Ball

In all of our experiments there are two agents: a *passer* accelerates as fast as possible towards a stationary ball in order to propel it between a *shooter* and the goal. The resulting speed of the ball is determined by the distance that the passer started from the ball. The shooter's task is to time its acceleration so that it intercepts the ball's path and redirects it into the goal. We constrain the shooter to accelerate at a fixed constant rate (while steering along a fixed line) once it has decided to begin its approach. *Thus the behavior to be learned consists of the decision of when to begin moving: at each action opportunity the shooter either starts or waits.* Once having started, the decision may not be retracted. The key issue is that the shooter must make its decision based on the observed field: the ball's and the shooter's (x, y, θ) coordinates reported at a (simulated) rate of $60Hz$.[1] The method in which the shooter makes this decision is called its *shooting policy*.

Throughout our experiments, the shooter's initial position varies randomly within a continuous range: its initial heading varies over 70 degrees and its initial x and y coordinates vary independently over 40 units as shown in Figure 2(a).[2] The two shooters pictured show the extreme possible starting positions, both in terms of heading and location.

Since the ball's momentum is initially across the front of the goal, the shooter must compen-

[1]The Dynamo system reported the coordinates at a rate of 60Hz using an overhead camera and color-coded objects [11].

[2]For reference, the width of the field (the side with the goal) shown in figure 2(a) is 480 units. The width of the goal is 80 units.

sate by aiming wide of the goal when making contact with the ball (see Figure 2(b)). Before beginning its approach, the shooter chooses a point wide of the goal at which to aim. Once deciding to start, it then steers along an imaginary line between this point and the shooter's initial position, continually adjusting its heading until it is moving in the right direction along this line (see Appendix B). The line along which the shooter steers in the *steering line*. The method in which the shooter chooses the steering line is called its *aiming policy*.

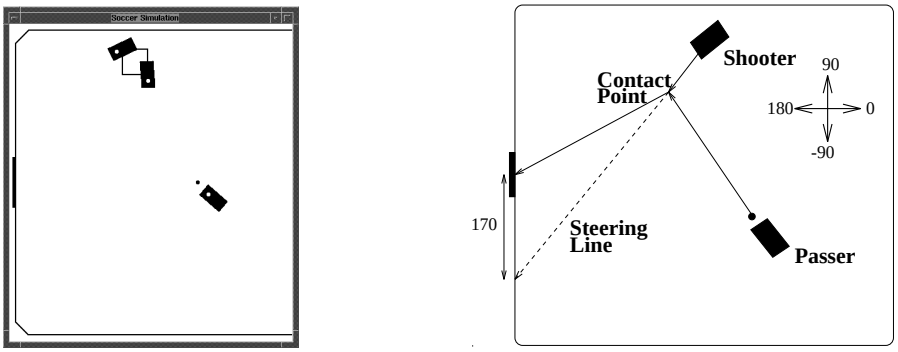


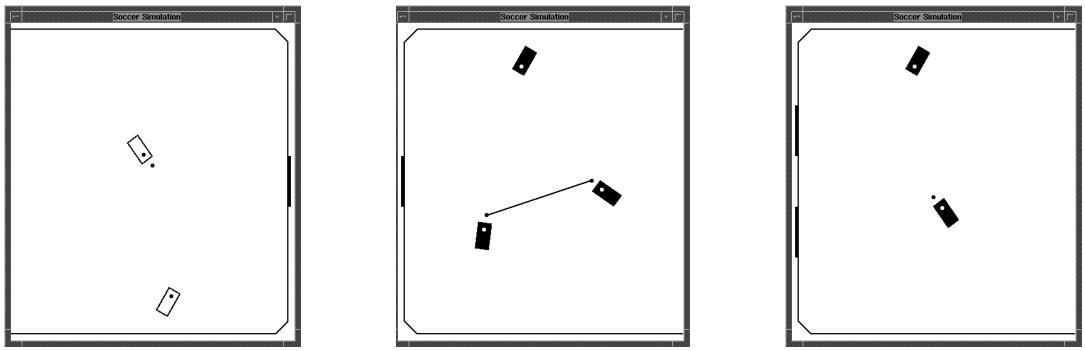
Figure 2: **(a)** The initial position for the experiments in this paper. The agent in the lower part of the picture, the *passer*, accelerates full speed ahead until it hits the ball. Another agent, the *shooter* then attempts to redirect the ball into the goal on the left. The two agents in the top of the figure illustrate the extremes of the range of angles of the shooter's initial position. The square behind these two agents indicates the range of the initial position of the *center* of the shooter. **(b)** A diagram illustrating the paths of the ball and the agents during a typical trial.

The task of learning a shooting policy has several parameters that can control its level of difficulty. First, the ball can be moving at the same speed for all training examples or at different speeds. Second, the ball can be coming with the same trajectory or with different trajectories. Third, the goal can always be in the same place during testing as during training, or it can change locations (think of this parameter as the possibility of aiming for different parts of the goal). Fourth, the training and testing can occur all in the same location, or the testing can be moved to a different *action quadrant*: a symmetrical location on the field. To acquire a robust behavior, we perform and report a series of experiments in which we increase the difficulty of our task incrementally. We develop a learning agent to test how far training in a limited scenario can extend. Table 1 indicates how the paper is organized according to the parameters varied throughout our experiments. Figure 3 illustrates some of these variations.

	Fixed	**Variable**
Ball Speed	Section 4	Sections 5, 6, 7
Ball Trajectory	Sections 4, 5	Sections 6, 7
Goal Location	Sections 4, 5, 6	Section 7
Action Quadrant	Sections 4, 5, 6, 7—training	Sections 4, 5, 6, 7—testing

Table 1: The parameters that control the difficulty of the task and the sections in which they are varied.

We use a supervised learning technique to learn the task at hand. Throughout the paper, all



(a) (b) (c)

Figure 3: Variations to the initial setup: **(a)** The initial position in the opposite corner of the field (a different action quadrant); **(b)** Varied ball trajectory: The line indicates the ball's possible initial positions, while the passer always starts directly behind the ball facing towards a fixed point. The passer's initial distance from the ball controls the speed at which the ball is passed; **(c)** Varied goal position: the placement of the higher and lower goals are both pictured at once.

training is done with simulated sensor noise of up to about 2 units for x and y and 2 degrees for θ. All reported success rates are based on at least 1000 trials.

4 Fixed Ball Motion

We began our experimentation with the ball always being passed with the same trajectory and the same speed for all training and testing examples. With this condition of *fixed ball motion*, the shooter could always aim at the same point wide of the goal, guaranteeing that if contact was made, the ball would be propelled in the right direction. That is to say, the shooter used a constant aiming policy. We determined that with the trajectory (140^o) and speed ($\approx$ 135 units/sec) of the ball we were initially using, the shooter would score when contacting the ball if its steering line was such that it aimed 170 units wide of the center of the goal (illustrated in Figure 2(b)). This point remains constant throughout this section and Section 5.

Before setting up any learning experiments, we found a simple fixed shooting policy that would allow the shooter to score consistently when starting at the exact center of its range of initial positions. Starting at this position, the shooter could score consistently if it began accelerating when the ball's distance to its projected point of intersection with the agent's path reached 110 units or less. We call this policy the *simple shooting policy*. However, this simple policy was clearly not appropriate for the entire range of shooter positions that we considered: when using this policy while starting at random positions, the shooter scored only 60.8% of the time.

4.1 Choosing the inputs

Convinced that a ML technique could provide a significantly better shooting policy, we decided to try using a NN as our initial attempt. We plan to experiment with other ML techniques on the same task in the future. We first considered how we should structure the NN in order to learn a

function from the current state of the world to an indication of whether the shooter should start accelerating or remain still and wait. The output of this function was fairly straightforward. It would indicate whether starting to accelerate in a world state described by the input values was likely to lead to a goal (outputs close to 1) or a miss (outputs close to 0). However, deciding how to represent the world state, i.e. the inputs to the NN, represented a core part of our research.

One option was to use coordinates (x, y, θ) for both the shooter and the ball. However, such inputs would not have generalized beyond the very limited training situation. Furthermore, they would have led to a higher dimensional function (6) than turned out to be necessary. Instead we chose to use just 3 easily-computable coordinate-independent predicates.

Since the line along which the agent steered was computed before it started moving (the line connecting the agent's initial position and the point 170 units wide of the goal), and since the ball's trajectory could be estimated (with some error due to noise) after getting two distinct position readings, the shooter was able to determine the point at which it hoped to strike the ball, or the *Contact Point*. It could then cheaply compute certain useful predicates:

- *Ball Distance*: the ball's distance to the Contact Point;
- *Agent Distance*: the agent's distance to the Contact Point; and
- *Heading Offset*: the difference between the agent's initial heading and its desired heading.

The physical meaning of these inputs is illustrated in Figure 4(a). These inputs proved to be sufficient for learning the task at hand. Furthermore, since they contained no coordinate-specific information, they enabled training in a narrow setting to apply much more widely as shown at the end of this section.

4.2 Gathering training data

Recall that when the shooter started in the center of its range, it would score using the simple shooting policy: it began moving when the Ball Distance was 110 units or less. However, to get a diverse training sample, we replaced this shooting policy with a *random shooting policy* of the form "at each opportunity, begin moving with probability $\frac{1}{x}$." To help choose x, we determined that the shooter had about 25 decision opportunities before the ball moved within 110 units of the Contact Point. Since we wanted the shooter to start moving before or after these 25 decision cycles with roughly equal probability so as to get a balanced training sample, we solved the equation $(\frac{x-1}{x})^{25} = .5 \Rightarrow x \approx 37$. Hence, when using the random shooting policy, the shooter started moving with probability 1/37 at each decision point.

Using this shooting policy, we then collected training data. Each instance consisted of four numbers: the three inputs (Ball Distance, Agent Distance, and Heading Offset) at the time that the shooter began accelerating and a 1 or 0 to indicate whether the shot was successful or not. A shot was successful only if it went directly from the front of the shooter into the goal as illustrated in Figure 2(b): a trial was halted unsuccessfully if the ball hit any corner or side of the shooter, or if the ball hit any wall other than the goal.

Running 2990 trials in this manner gave us sufficient training data to learn to shoot a moving ball into the goal. The success rate using the random shooting policy was 19.7%. In particular, only 590 of the training examples were positive instances.

4.3 Training

Using this data, we were able to train a NN that the shooter could use as a part of a learned shooting policy that enabled it to score consistently. We tried several configurations of the NN, settling on one with a single layer of 2 hidden units and a learning rate of .01. Each layer had a bias unit with constant input of 1.0. We normalized all other inputs to fall roughly between 0.0 and 1.0 (see Figure 4(b)). The target outputs were .9 for positive examples (successful trials) and .1 for negative examples. Weights were all initialized randomly between -0.5 and 0.5. The resulting NN is pictured in Figure 4(b). This NN was not the result of an exhaustive search for the optimal configuration to suit this task, but rather the quickest and most successful of about 10 alternatives with different numbers of hidden units and different learning rates. Once settling on this configuration, we never changed it, concentrating instead on other research issues.

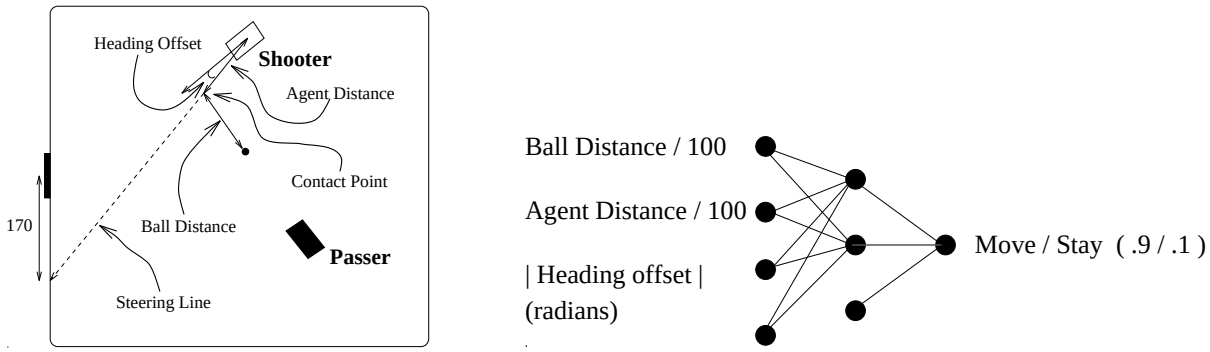


Figure 4: **(a)** The predicates we used to describe the world for the purpose of learning to shoot a moving ball are illustrated. **(b)** The NN used to learn the shooting policy. The NN has 2 hidden units with a bias unit at both the input and hidden layers.

Training this NN on the entire training set for 3000 epochs resulted in a mean squared error of .0386 and 253 of the examples misclassified (i.e. the output was closer to the wrong output than the correct one). Training for more epochs did not help noticeably. Due to the sensor noise during training, the concept was not perfectly learned.

4.4 Testing

After training, a NN could be used very cheaply—just a single forward pass at each decision point—to decide when to begin accelerating. Notice that within a single trial, only one input to the NN varied: the Ball Distance decreased as the ball approached the Contact Point. Thus, the output of the NN tended to vary fairly regularly. As the ball began approaching, the output began increasing first slowly and then sharply. After reaching its peak, the output began decreasing slowly at first and then sharply. The optimal time for the shooter to begin accelerating was at the peak of this function, however since the function peaked at different values on different trials, we used the following *3-input NN shooting policy*:

- Begin accelerating when Output $\geq$ .6 AND Output $<$ Previous output - .01.

Requiring that Output $\geq$ .6 ensured that the shooter would only start moving if it "believed" it was more likely to score than to miss. Output $<$ Previous output - .01 became true when the

output of the NN was just past its peak. Requiring that the output decrease by at least .01 from the previous output ensured that the decrease was not due simply to sensor noise.

Using this learned 3-input NN shooting policy, the shooter scored **96.5%** of the time. The results reported in this section are summarized in Table 2.

Initial Shooter Position	Shooting Policy	Success (%)
Constant	Simple	100
Varying	Simple	60.8
Varying	Random	19.7
Varying	3-input NN	**96.5**

Table 2: Results before and after learning for fixed ball motion.

Even more important than the high success rate achieved when using the learned shooting policy was the fact that the shooter achieved the same success rate in each of the four symmetrical reflections of the training situation (the four *action quadrants*). **With no further training, the shooter was able to score from either side of the goal on either side of the field.** Figure 3(a) illustrates one of the three symmetrical scenarios. The world description used as input to the NN contained no information specific to the location on the field, but instead captured only information about the relative positions of the shooter, the ball, and the goal. Thanks to these flexible inputs, training in one situation was applicable to several other situations.

5 Varying the Ball's Speed

Encouraged by the success and flexibility of our initial solution, we next varied a parameter of the initial setup to test if the solution would extend further. Throughout Section 4, the passer started 35 units away from the ball and accelerated full-speed ahead until striking it. This process consistently propelled the ball at about 135 units/sec. To make the task more challenging, we varied the ball's speed by starting the passer randomly within a range of 32–38 units away from the ball. Now the ball would travel towards the shooter with a speed of 110–180 units/sec.

Before making any changes to the shooter's shooting policy, we tested the policy trained in Section 4 on the new task. However, we found that the 3-input NN was not sufficient to handle the varying speed, giving a success rate of only 49.1% due to the mistiming of the acceleration. In order to accommodate for this mistiming, we added a fourth input to the NN in order to represent the speed of the ball: *Ball Speed*

The shooter computed the Ball Speed from the ball's change in position over a given amount of time. Due to sensor noise, the change in position over a single time slice did not give an accurate reading. On the other hand, since the ball slowed down over time, the shooter could also not take the ball's total change in position over time. As a compromise, the shooter computed the Ball Speed from the ball's change in position over the last 10 time slices (or fewer if 10 positions had not yet been observed).

To accommodate for the additional quantity used to describe a world state, we gathered new training data. As before, the shooter used the random shooting policy during the training trials. This time, however, each training instance consisted of *four* inputs describing the state of

the world plus an output indicating whether or not the trial was a successful one. Of the 5737 samples gathered for training, 963—or 16.8%—were positive examples.

For the purposes of training the new NN, we scaled the Ball Speed of the ball to fall between 0.0 and 1.0: $\frac{speed-110}{70}$. Except for the fourth input, the new NN looked like the one pictured in Figure 4(b). It had two hidden units, a bias unit at each level, and a learning rate of .001. Training this NN for 4000 epochs resulted in a mean squared error of .0512 with 651 of the instances misclassified.

Using this new NN and the same decision function over its output as before, or the *4-input NN shooting policy*, our shooter was able to score **91.5%** of the time with the ball moving at different speeds. **Again, this success rate was observed in all four action quadrants.**

Shooting Policy	Success (%)
3-input NN	49.1
Random	16.8
4-input NN	**91.5**

Table 3: When the Ball Speed varies, an additional input is needed.

6 Varying the Ball's Trajectory

To this point, we have shown that the inputs we used to train our NN have allowed training in a particular part of the field to apply to other parts of the field. In this section we describe experiments that show that the 4-input NN shooting policy described in Section 5 is flexible still further. **All experiments in this section use exactly the same shooting policy with no further retraining.**

Since the inputs are all relative to the Contact Point, we predicted that the trajectory with which the ball approached the shooter would not affect the performance of the NN adversely. In order to test this hypothesis we changed the initial positions of the ball and the passer so that the ball would cross the shooter's path with a different heading (90^o as opposed to 140^o). This variation in the initial setup is illustrated in Figure 5. The ball's speed still varied as in Section 5.

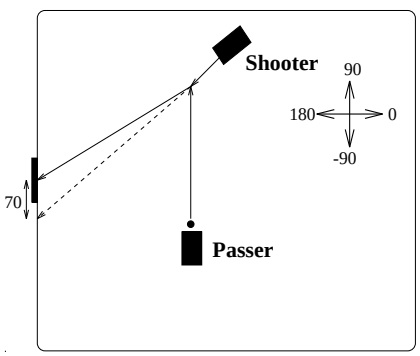


Figure 5: When the ball approached with a trajectory of 90^o instead of 140^o, the shooter had to change its aiming policy to aim 70 units wide of the goal instead of 170 units wide.

Indeed, the shooter was able to consistently make contact with the ball and redirect it towards the goal. However, it never scored because its steering line was aiming it too wide of the goal. Due to the ball's changed trajectory, aiming 170 units wide of the goal was no longer an appropriate aiming policy.

With the ball always coming with the same trajectory, we could simply change the steering line such that the shooter was aiming 70 units wide of the center of the goal. Doing so led to a success rate of **96.3%**—even better than the actual training situation. This improved success rate can be accounted for by the fact that the ball was approaching the agent more directly so that it was slightly easier to hit. Nonetheless, it remained a difficult task, and it was successfully accomplished with no retraining. **Not only did our learned shooting policy generalize to different areas of the field, but it also generalized to different ball trajectories.**

Having used the same shooting policy to successfully shoot balls moving at 2 different trajectories, we knew that we could vary the ball's trajectory over a continuous range and still use the same policy to score. The only problem would be altering the shooter's aiming policy. Thus far we had chosen the steering line by hand, but we did not want to do that for every different possible trajectory of the ball. In fact, this problem gave us the opportunity to put the principal espoused by this paper into use once again.

For the experiments described in the remainder of this section, the ball's initial trajectory ranged randomly over 63^o ($82^o - 145^o$). Figure 3(b) illustrates this range. The policy used by the shooter to decide when to begin accelerating was exactly the same one learned in Section 5 with no retraining of the NN. On top of this NN, we added a new one to determine the direction in which the shooter should steer. The steering line determined by this direction and the shooter's current position could then be used to compute the predicates needed as input to the original NN.

For this new NN we again chose parameters that would allow it to generalize beyond the training situation. The new input we chose was the angle between the ball's path and the line connecting the shooter's initial position with the center of the goal: the *Ball-Agent Angle*. The output was the *Angle Wide* of this second line that the shooter should steer. These quantities are illustrated in Figure 6(a). We found that performance improved slightly when we added as a second input an estimate of the ball's speed. However, since the NN in the 4-input NN shooting policy could only be used after the shooter determined the steering line, we used an estimate of Ball Speed after just two position readings. Then, the shooter's steering line was chosen by the learned aiming policy and the 4-input NN shooting policy could be used to time the shot.

To gather training data for the new NN, we ran many trials with the ball passed with different speeds and different trajectories while the Angle Wide was set to a random number between 0.0 and 0.4 radians. Using only the 783 positive examples, we then trained the NN to be used for the shooters' learned aiming policy. Again, the inputs and the outputs were scaled to fall roughly between 0.0 and 1.0, there were two hidden units and two bias units, the weights were initialized randomly, and the learning rate was .01. The resulting NN is pictured in Figure 6(b). Training this NN for 4000 epochs gave a mean squared error of .0421.

Using this *2-input NN aiming policy* to decide where to aim and the old 4-input NN shooting policy to decide when to accelerate, the shooter scored **95.4%** of the time while the ball's speed and trajectory varied. Using a NN with just 1 input (omitting the speed estimate) and one hidden unit, or the *1-input NN aiming policy*, gave a success rate of 92.8%. Being satisfied with

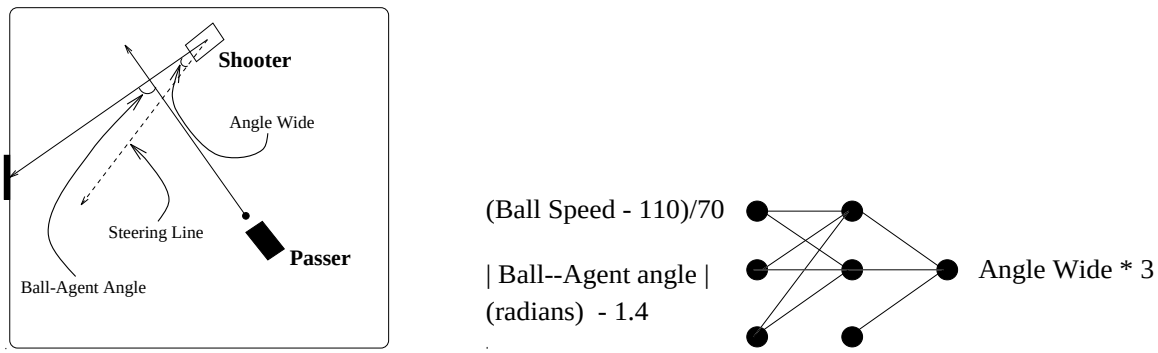


Figure 6: **(a)** The predicates we used to describe the world for the purpose of learning an aiming policy. **(b)** The NN used to learn the aiming policy. The NN has 2 hidden units with a bias unit at both the input and hidden layers.

this performance, we did not experiment with other NN configurations.

7 Moving the Goal

To test if the learned aiming policy could generalize beyond its training situation, we then moved the goal by a goal-width both down and up along the same side of the field (see Figure 3(c)). One can think of this variation as the shooter aiming for different parts of a larger goal. Changing nothing but the shooter's knowledge of where the goal was located, the shooter scored **84.1%** of the time on the lower goal and **74.9%** of the time on the upper goal. The discrepancy between these values can be explained by the greater difficulty of shooting a ball in a direction close to the direction that it was already travelling. As one would expect, when the initial situation was flipped so that the shooter began in the lower quadrant, these two success rates also flipped: the shooter scored more frequently when shooting at the higher goal. Notice that in this case, the representation of the output was as important as the inputs for generalization beyond the training situation.

Aiming Policy	Goal Position	Success (%)
1-input NN	Middle	92.8
2-input NN	Middle	**95.4**
2-input NN	Lower by one goal-width	**84.1**
2-input NN	Higher by one goal-width	**74.9**

Table 4: When the ball's trajectory varied, a new aiming policy was needed. Results are reasonable even when the goal is moved to a new position.

8 Discussion and Conclusion

The experiments described in this paper demonstrate the power of using flexible inputs to learning algorithms. For two different learned policies, we achieved good performance in situations much

broader than their training situations. Most important was the generalization of the the shooter's shooting policy. Trained in one action quadrant with the goal in a fixed place and the ball moving with a fixed trajectory, it was used successfully in all four action quadrants with the ball moving at different trajectories and with the goal in different locations. The shooter's aiming policy also generalized to the three action quadrants that were not used during training as well as to different goal positions.

The secret behind the flexibility of these learned policies was that their inputs (and outputs) were coordinate-independent. All of the measurements represented the relative positions of the ball, the shooter and the goal, with no mention of x, y,or θ. Had we used the coordinates as inputs, we would have had to train new NNs for every new situation. We may have been able to find a single NN that could learn a disjunctive concept with more variables, but we would have had to at least provide new training data to cover the new situations.

By using these flexible inputs, we can incrementally teach general behavior templates to our robots, building them on top of each other. Similar to human soccer players, they can first learn to make contact with a moving ball, then learn to aim it, and only then start thinking about trying to beat an opponent and about team-level strategies.

Our future research agenda includes both of these higher level challenges. We will collect several different behaviors learned in a similar manner to the shooting described in this paper. Then we will use them to plan how to work as a part of a team against hostile opponents. We will put our robots through an intense training regimen to prepare them for RoboCup '97.

Acknowledgements

This research is sponsored by the Wright Laboratory, Aeronautical Systems Center, Air Force Materiel Command, USAF, and the Advanced Research Projects Agency (ARPA) under grant number F33615-93-1-1330. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Wright Laboratory or the U. S. Government.

References

[1] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda. Purposive behavior acquisition on a real robot by vision-based reinforcement learning. In *Proc. of MLC-COLT (Machine Learning Confernce and Computer Learning Theory) Workshop on Robot Learning*, pages 1–9, 1994.

[2] M. Asada, E. Uchibe, S. Noda, S. Tawaratsumida, and K. Hosoda. Coordination of multiple behaviors acquired by vision-based reinforcement learning. In *Proc. of IEEE/RSJ/GI International Conference on Intelligent Robots and Systems 1994 (IROS '94)*, pages 917–924, 1994.

[3] Rod Barman, Stewart Kingdon, Alan Mackworth, Kinesh Pai, Michael Sahota, Heath Wilkinson, and Ying Zhang. Dynamite: A testbed for multiple mobile robots. In *Proceedings of the 1993 IJCAI Workshop on Dynamically Interacting Robots*, Chambery, France, 1993. IJCAII.

[4] Roger Ford, Craig Boutilier, and Keiji Kanazawa. Exploiting natural structure in reinforcement learning: Experience in robot soccer-playing, 1994. Unpublished Manuscript.

[5] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. Robocup: The robot world cup initiative. In *IJCAI-95 Workshop on Entertainment and AI/Alife*, Montreal, Quebec, August 1995.

[6] Jean-Claude Latombe. A fast path planner for a car-like indoor mobile robot. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 659–665, 1991.

[7] Dean A. Pormerleau. *Neural Network Perception for Mobile Robot Guidance*. Kluwer Academic Publishers, 1993.

[8] William H. Press. *Numerical Recipes in C: the art of scientific computing*, pages 710–713. Cambridge University Press, Cambridge, 1988.

[9] J. A. Reeds and R. A. Shepp. Optimal paths for a car that goes both forward and backward. *Pacific Journal of Mathematics*, 145(2):367–393, 1991.

[10] Michael Sahota. Personal correspondence, August 1994.

[11] Michael K. Sahota. Real-time intelligent behaviour in dynamic environments: Soccer-playing robots. Master's thesis, University of British Columbia, August 1993.

[12] O. G. Selfridge, R. S. Sutton, and A. G. Barto. Training and tracking in robotics. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 670–672, Los Angeles, CA, 1985. Morgan Kaufman.

A Simulator Physics

The physics of our simulator are based largely on those embedded in the simulator used by the the Dynamo group [11] with some minor adjustments. The ball has a radius of 5.0 units, mass of 0.333 units, and a drag of 30 units/sec^2. This value of drag is typical for a ping-pong ball (30 cm/sec^2).

The agents are all 40.0 units long by 20.0 units wide with a mass of 10.0 units. The agent's position is updated according to the steering and throttle commands it receives using the fourth-order Runge-Kutta formula [8].

Collisions between objects in the world are handled as follows. The ball bounces off walls elastically and stops if it enters a goal, while agents stop if they collide with one another or with a wall. Collisions between the ball and an agent take into account momenta of the colliding objects as well as the corners on the agents.

B Line Following

For the clients in this paper, the only path planning needed is the ability to steer along a straight line. Given a line and periodically it's own position, the client's goal is to eventually be travelling precisely parallel to the line within a certain distance from the line. This distance, or the *tolerance*, is a parameter, but throughout this paper it is set to 5 units. Since the agent's width is 20 units, this tolerance allows the client to strike the ball as expected.

The client begins by computing its distance from the line as well as its offset from the correct heading (the slope of the line). If the client's distance is greater than the tolerance, then the agent steers sharply towards the line. In particular, it sets its steering in proportion to its distance from the line minus the

tolerance. If the distance is no more than the tolerance (or once it has become so), then the client steers towards the line with a force proportional to its heading offset. Thus, when the client is heading parallel to the line, its wheel is centered.

Although this method does not force the client to travel exactly along the given line, it does allow the client to be moving in precisely the right direction. Thus it is able to reliably strike the ball in a particular direction.