Beating a Defender in Robotic Soccer: Memory-Based Learning of a Continuous Function

Peter Stone and Manuela Veloso

December 11, 1995
CMU-CS-95-222

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Learning how to adjust to an opponent's position is critical to the success of having intelligent agents collaborating towards the achievement of specific tasks in unfriendly environments. This paper describes our work on developing methods to learn to choose an action based on a continuous-valued state attribute indicating the position of an opponent. We use a framework in which teams of agents compete in a simulator of a game of robotic soccer. We introduce a memory-based supervised learning strategy which enables an agent to choose to pass or shoot in the presence of a defender. In our memory model, training examples affect neighboring generalized learned instances with different weights. We conduct experiments in which the agent incrementally learns to approximate a function with a continuous domain. Then we investigate the question of how the agent performs in nondeterministic variations of the training situations. Our experiments indicate that when the random variations fall within some bound of the initial training, the agent performs better with some initial training rather than from a tabula-rasa.

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, Memory-Based Learning, Incremental Learning, Adaptive Learning, Continuous-function Learning

1 Introduction

One of the ultimate goals subjacent to the development of intelligent agents is to have multiple agents collaborating in the achievement of specific tasks in the presence of hostile opponents. Our research works towards this broad goal from a Machine Learning perspective. We are particularly interested in investigating how an individual intelligent agent can choose an action in an adversarial environment. We assume that the agent has a specific goal to achieve. We conduct this investigation in a framework where teams of agents compete in a game of robotic soccer. The real system of model cars remotely controlled from off-board computers is 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 [5]. In particular, our simulator's code is adapted from their own code, for which we thank Michael Sahota whose work [8] and personal correspondence [7] has been motivating and invaluable. The simulator facilitates the control of any number of cars 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. 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 shows the simulator graphics.

We have focused on the question of learning to choose among actions in the presence of an adversary. This paper describes our work on applying memory-based supervised learning techniques to acquire strategy knowledge that enables an agent to decide how to achieve a goal. For other work in the same domain, please see [12, 13].

The input to the learning task includes a continuous-valued range of the position of the adversary. This raises the question of how to discretize the space of values into a set of learned features. We present our empirical studies and results on learning an appropriate generalization degree in this continuous-valued space. Due to the cost of learning and reusing a large set of specialized instances, we notice a clear advantage to having an appropriate degree of generalization.

Next, we address the issue of the effect of differences between past episodes and the current situation. We performed extensive experiments, training the system under particular conditions and then testing it (with training continuing incrementally) in nondeterministic variations of the training situation. Our results show that when the random variations fall within some bound of the initial training, the agent performs better with some initial training rather than from a tabula-rasa. This intuitive fact is interestingly well- supported by our empirical results.

The paper is organized in five sections. Section 2 describes the memory-based learning method and the experimental setup. Section 3 presents and discusses the results obtained. Section 4 discusses related work and our own directions for future work. Section 5 summarizes our conclusions.

2 Learning Method

The learning method we develop here applies to an agent trying to learn a function with a continuous domain. We situate the method in the game of robotic soccer.

2.1 Motivation

Imagine yourself on a soccer field with the ball at your feet, 25 yards from the opponents' goal. There is a single defender (the goalie) between you and the goal, and a teammate of yours is off to one side, unmarked, and ready to receive a pass. Imagine also, if you must, that you are a skilled soccer player, and that you enjoy playing the game immensely. You would like nothing more than to help your team score a goal.

In this situation, you have two options: shoot the ball directly at the goal or pass the ball to your teammate so that she can take a shot. The first time you find yourself in this situation, you do not know what to do. You randomly decide to shoot or to pass and you take note of the result. As you continue to play, however, you find yourself back in the same position repeatedly. You are in the same position with the ball, and you have a teammate in the same position ready to receive a pass. The only difference is that the defender's starting position is never quite the same. Furthermore, since you and your teammate

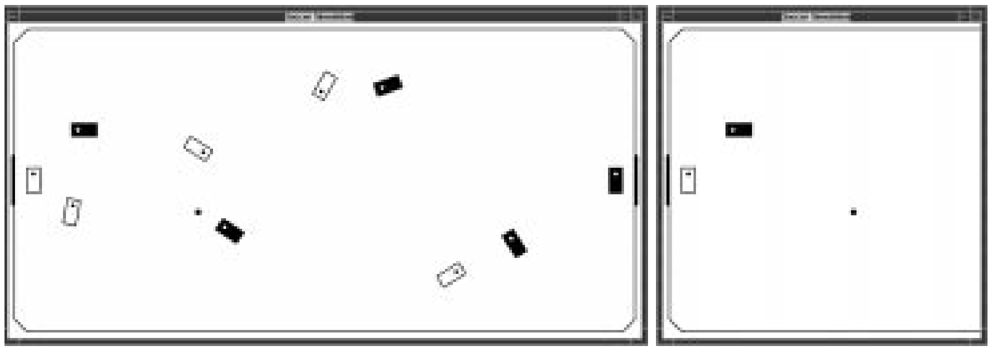

Figure 1: On the left is the graphic view of our simulator. On the right is the initial position for all of the experiments in this paper. The teammate (black) remains stationary, the defender (white) moves in a small circle at different speeds, and the ball can move either directly towards the goal or towards the teammate. The position of the ball represents the position of the learning agent.

are remarkably consistent in your abilities, the ball travels along the same path every time you shoot; even when you pass the ball, your teammate redirects it in the same way every time. That is, the ball's motion is completely deterministic once you have either shot or passed it.

In contrast, the defender is continuously moving in front of the goal, and his motion may or may not be deterministic. At first you shoot or pass randomly, but after a few successful attempts, you begin to learn, based on the defender's position, whether to shoot or to pass. Even if the defender's motion is in fact deterministic, you can never be entirely sure of scoring unless you have scored before when the defender started in *exactly* the same position.

2.2 Experimental Setup

We simulate this situation by placing a ball and a stationary car acting as the "teammate" in specific places on the field. Then we place another car, the "defender," in front of the goal. The defender moves in a small circle in front of the goal at some speed and begins at some random point along this circle. The learning agent ("you") must take one of two possible actions: *shoot* straight towards the goal, or *pass* to the teammate so that the ball will rebound towards the goal. A snapshot of the experimental setup is shown graphically in Figure 1.

The task is essentially to learn two functions, each with one continuous input variable, namely the defender's position. Based on this position, which can be represented unambiguously as the angle at which it is facing, ϕ, the agent tries to learn the probability of scoring when shooting, $P_s^*(\phi)$, and the probability of scoring when passing, $P_p^*(\phi)$.[1] If these functions were learned completely, which would only be possible if the defender's motion were deterministic, then both functions would be binary partitions: $P_s^*, P_p^* : [0.0, 360.0) \mapsto \{-1, 1\}$.[2] That is, the agent would know without doubt for any given ϕ whether a shot, a pass, both, or neither would achieve its goal. However, since the agent cannot have had experience for every possible ϕ, and since the defender may not move at the same speed each time, the learned functions must be approximations: $P_s, P_p : [0.0, 360.0) \mapsto [-1.0, 1.0]$.

In order to enable the agent to learn approximations to the functions P_s^* and P_p^*, we gave it a memory in which it could store its experiences and from which it could retrieve its current approximations $P_s(\phi)$ and $P_p(\phi)$. We explored and developed appropriate methods of storing to and retrieving from memory and an algorithm for deciding what action to take based on the retrieved values.

[1] As per convention, P^* represents the target (optimal) function.

[2] Although we think of P_s^* and P_p^* as functions from angles to probabilities, we will use -1 rather than 0 as the lower bound of the range. This representation simplifies many of our illustrative calculations.

2.3 Memory Model

Storing every individual experience in memory would be inefficient both in terms of amount of memory required and in terms of generalization time. Therefore, we store P_s and P_p only at discrete, evenly-spaced values of ϕ. That is, for a memory of size M (with M dividing evenly into 360 for simplicity), we keep values of $P_p(\theta)$ and $P_s(\theta)$ for $\theta \in \{360n/M \mid 0 \leq n < M\}$. We store memory as an array "**Mem**" of size M such that $\mathbf{Mem}[n]$ has values for both $P_p(360n/M)$ and $P_s(360n/M)$. Using a fixed memory size precludes using memory-based techniques such as K-Nearest-Neighbors (kNN) and kernel regression which require that every experience be stored, choosing the most relevant only at decision time. Most of our experiments were conducted with memories of size 360 (low generalization) or of size 18 (high generalization), i.e. $M = 18$ or $M = 360$. As will be seen from our results, the memory size had a large effect on the rate of learning.

2.3.1 Storing to Memory

With M discrete memory storage slots, the problem then arises as to how a specific training example should be generalized. Training examples are represented here as $E_{\phi,a,r}$, consisting of an angle ϕ, an action a, and a result r where ϕ is the initial position of the defender, a is "s" or "p" for "shoot" or "pass," and r is "1" or "-1" for "goal" or " miss" respectively. For instance, $E_{72.345,p,1}$ represents a pass resulting in a goal for which the defender started at position $72.345°$ on its circle.

The most straightforward technique would be to store the result at the single memory slot whose index is closest to ϕ, i.e., round ϕ to the nearest θ for which $\mathbf{Mem}[\theta]$ is defined, and then set $P_a(\theta) = r$. However, this technique does not provide for the case in which we have two training examples $E_{\phi_1,a,1}$ and $E_{\phi_2,a,-1}$, where ϕ_1 and ϕ_2 both round to the same θ. In particular, there is no way of scaling how indicative $E_{\phi,a,r}$ is of $P_a(\theta)$.

In order to combat this problem, we scale the value stored to $P_a(\theta)$ by the inverse of the distance between ϕ and θ relative to the distance between memory indices. A result r at a given ϕ is multiplied by $1 - \frac{|\phi-\theta|}{360/M}$ before being stored to $\mathbf{Mem}[\theta]$. In this way, training examples with ϕ's that are closer to θ can affect $\mathbf{Mem}[\theta]$ more strongly. For example, with $M = 18$ (so $\mathbf{Mem}[\theta]$ is defined for $\theta = 20n$), $E_{116.5,p,-1}$ causes $P_p(120)$ to be updated by a value of $-1 * (1 - \frac{3.5}{20}) = -.825$. Call this our *basic* memory storage technique:

Basic Memory Storage of $E_{\phi,a,r}$ in $\mathbf{Mem}[\theta]$
• $r' = r * (1 - \frac{|\phi-\theta|}{360/M})$.
• If $|r'| \geq |P_a(\theta)|$ Then $P_a(\theta) = r'$.

Using this generalization function, the "update" of $P_p(120)$ would only have an effect at all if $|P_p(120)| \leq .825$ prior to this training example. Consequently, only the past training example for action a with ϕ closest to θ is reflected in $P_a(\theta)$: presumably, this training example is most likely to accurately predict $P^*_a(\theta)$. Notice that this basic memory storage technique is appropriate when the defender's motion is deterministic. In order to handle variations in the defender's speed, we introduce later a more complex memory storage technique. The method of scaling a result based on the difference between ϕ and θ will remain unchanged.

In our example above, $E_{116.5,p,-1}$ would not only affect $\mathbf{Mem}[120]$: as long as $|P_p(100)|$ was not already larger, its value would be set to $-1 * (1 - \frac{16.5}{20}) = -.175$. Notice that any training example $E_{\phi,p,r}$ with $83.5 \leq \phi \leq 116.5$ could override this value. Since 116.5 is so much closer to 120 than it is to 100, it makes sense that $E_{116.5,a,r}$ affects $\mathbf{Mem}[120]$ more strongly than it affects $\mathbf{Mem}[100]$. However, $E_{110,a,r}$ would affect both memory values equally. This memory storage technique is similar to the kNN and kernel regression function approximation techniques which estimate $f(\phi)$ based on $f(\theta)$ possibly scaled by the distance from θ to ϕ for the k nearest values of θ. In our linear continuum of defender position, our memory generalizes training examples to the 2 nearest memory locations.[3]

[3]For particularly large values of M it is useful to generalize training examples to more memory locations, particularly at the early stages of learning. However for the values of M considered in this paper, we always generalize to the 2 nearest memory locations.

2.3.2 Retrieving from Memory

Since individual training examples affect multiple memory locations, we use a simple technique for retrieving $P_a(\phi)$ from memory when deciding whether to shoot or to pass. We round ϕ to the nearest θ for which $\mathbf{Mem}[\theta]$ is defined, and then take $P_a(\theta)$ as the value of $P_a(\phi)$. Thus, each $\mathbf{Mem}[\theta]$ represents $P_a(\phi)$ for $\theta - 360/2M \leq \phi < \theta + 360/2M$. Notice that retrieval is much simpler when using this technique than when using kNN or kernel regression: we look directly to the closest fixed memory position, thus eliminating the indexing and weighting problems involved in finding the k closest training examples and (possibly) scaling their results. We used this retrieval technique throughout our experiments, concentrating the trickiness of our function learning at storage time.

2.4 Choosing an Action

The action selection method is designed to make use of memory to select the action most probable to succeed, and to fill memory when no useful memories were available. For example, when the defender is at position ϕ, the agent begins by retrieving $P_p(\phi)$ and $P_s(\phi)$ as described in Section 2.3.2. Then, it acts according to the following function:

```
If $P_p(\phi) = P_s(\phi)$ (no basis for a decision), shoot or pass randomly.
else If $P_p(\phi) > 0$ and $P_p(\phi) > P_s(\phi)$, pass.
    else If $P_s(\phi) > 0$ and $P_s(\phi) > P_p(\phi)$, shoot.
        else If $P_p(\phi) = 0$, (no previous passes) pass.
            else If $P_s(\phi) = 0$, (no previous shots) shoot.
                else $(P_p(\phi), P_s(\phi) < 0)$ shoot or pass randomly.
```

An action is only selected based on the memory values if these values indicate that one action is likely to succeed and that it is better than the other. If, on the other hand, neither value $P_p(\phi)$ nor $P_s(\phi)$ indicate a positive likelihood of success, then an action is chosen randomly. The only exception to this last rule is when one of the values is zero,[4] suggesting that there has not yet been any training examples for that action at that memory location. In this case, there is a bias towards exploring the untried action in order to fill out memory.

3 Experiments and Results

In this section, we present the results of our experiments. We begin by finding an appropriate memory size to use for this task. Then we explore our agent's ability to learn time-varying and nondeterministic defender behavior, introducing a more sophisticated memory storage technique.

While examining the results, keep in mind that even if the agent used the functions P^*_s and P^*_p to decide whether to shoot or to pass, the success rate would be significantly less than 100% (it would differ for different defender speeds): there were many defender starting positions for which neither shooting nor passing led to a goal (see Figure 2). For example, from our experiments with the defender moving at a constant speed of

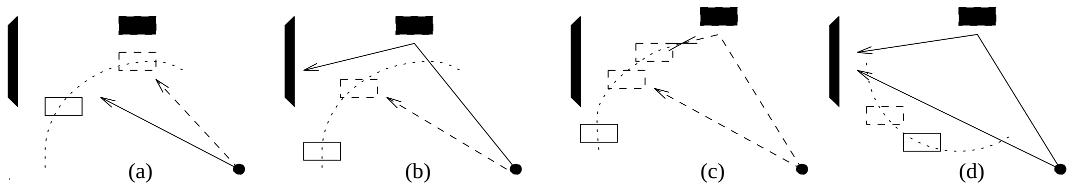

Figure 2: For different defender starting positions (solid rectangle), the agent can score when a) shooting, b) passing, c) neither, or d) both.

[4]Recall that a memory value of 0 is equivalent to a probability of .5, representing no reason to believe that the action will succeed or fail.

50,[5] we found that an agent acting optimally scores 73.6% of the time; an agent acting randomly scores only 41.3% of the time. These values set good reference points for evaluating our learning agent's performance. We indicate the scoring rate of an optimally acting agent on our graphs.

In order to increase the optimal success rate, we also experimented with allowing the agent not to act when $P_p, P_s < 0$ for a given defender position, i.e. when neither shooting nor passing was likely to work. The agent then scored 100% of the time by *waiting* until the defender moved into a position in which scoring was possible. In this setup, however, we had a difficult time collecting meaningful results, since the agent learned how to score when the defender was in a single position and then only acted when it was near that position. Therefore, we required the agent to act immediately.

3.1 Memory Size

Our first endeavor was to find an appropriate memory size for our task. Memory size corresponds to degree of generalization of training data. We ran experiments with the defender moving at two different speeds and with the agent having various memory sizes. Table 1 presents the results.

	Success Rate		Trials To Reach 70%	
M	Spd. 10	Spd. 50	Spd. 10	Spd. 50
1	44.1	35.7	-	-
2	53.2	57.4	-	-
3	70.6	66.1	155	-
4	60.8	68.7	-	100
5	65.2	69.3	-	100
6	61.4	75.5	-	100
8	67.5	74.1	-	100
9	68.8	72.2	-	448
10	64.4	69.0	-	100
12	68.5	72.0	100	100
18	69.7	74.9	126	100
20	71.9	71.1	297	139
24	67.3	74.7	219	100
30	72.9	74.5	166	319
36	73.7	75.1	153	189
40	74.4	74.4	319	315
45	72.8	76.0	465	362
60	71.4	76.2	461	100
180	75.8	74.3	655	821
360	73.5	74.0	1034	1080

Table 1: This table shows success rate during 1000 test trials after 1000 trials of training as well as the number of training trials it took to reach a success rate of 70% overall (after at least 100 trials). Trials that never reached the 70% mark are marked as "-". The ideal memory size would exhibit a speed-independent high success rate and quick learning rate (low number of trials to reach 70% success). The smaller memories tend to learn quicker but not as well, while the larger memories learn better but more slowly.

These numbers show the tradeoff between better performance in the long run (larger memory), and quicker learning (smaller memory). Smaller memories enable quicker learning due to the fact that the retrieval slots are fixed. Notice that there is no memory size that is absolutely better than all others, but there are certainly some sizes that are worse. From these trials we determined that, for this particular task, a memory of size 18 is dense enough to give good performance and sparse enough to learn relatively quickly. For the remainder of our experiments, we ran trials both with memories of size 18 and memories of size 360 in order to further compare the results. Repeatedly, we found that the smaller, more generalized memory gave faster learning without sacrificing much long-term performance.

3.2 Adaptive Memory

Imagine yourself again as the agent in this setup. Suppose that you have a very important match to play tomorrow and in your diligent way, you decide to learn for a single ball position whether it is best to pass or to shoot. You convince your teammate to come and stand in the place where you plan to pass, and then you convince your goalie to start in several different positions for two attempts: one shot and one pass. Of course the goalie will move to try to block the ball, but being a consistent goalie, you know that she will always move in the same way. Therefore you need only try shooting and passing once for each starting position. After a short amount of time, you have learned perfectly whether you should shoot or pass when the goalie is in a given position (with only a little error due to the necessity of rounding the goalie's position to the nearest position used for training).

[5] In the simulator, "50" represents 50 cm/s. We omit the units in the remainder of the paper.

The next day you turn up at the big game confident that if the ball is in your chosen spot, you will be able to choose correctly whether to shoot or to pass. You know that the opposing goalie is just as consistent as your own, so you believe that everything you learned yesterday should apply. But alas, your first attempt at a shot—one that you were sure would score—is blocked by the opposing goalie. What happened? The goalie's behavior is still deterministic, but it has changed completely: the new goalie is *slower* than your own. If you keep acting based on your experiences in practice, you are not going to score much, so you had better start adapting your memory to the current situation.

The memory-based technique we have been using so far works well when the defender's motion is deterministic and remains unchanged over time. However, if some noise is added to the defender's motion or if the defender changes its speed over time, then we need to use a more powerful technique. The technique we have used to this point converges monotonically since it assumes that once $P^*_a(\theta)$ has been learned perfectly at a memory location, then P_a need never change. If there is a training example $E_{\theta,a,-1}$, then no number of nearby conflicting examples $E_{\theta+.0001,a,1}$ will alter the value in $\mathbf{Mem}[\theta]$.

In our current scenario, memory needs to be able to *adapt* in response to new conflicting examples. In order to accommodate this requirement, we change our method of storing experiences to memory. We continue to scale the result of an experience $E_{\phi,a,r}$ stored to $\mathbf{Mem}[\theta]$ by $1 - \frac{|\phi-\theta|}{360/M}$, but rather than only storing the result of the experience with ϕ closest to θ, we now let each experience with $\theta - 360/2M \leq \phi < \theta + 360/2M$ affect $\mathbf{Mem}[\theta]$ in proportion to the distance $|\theta - \phi|$. In particular, $\mathbf{Mem}[\theta]$ keeps running sums of the magnitudes of scaled results, $\mathbf{Mem}[\theta]$.*total-a-results*, and of scaled positive results, $\mathbf{Mem}[\theta]$.*positive-a-results*, affecting $P_a(\theta)$, where "a" stands for "s" or "p" as before. Then at any given time, $P_a(\theta) = -1 + 2 * \frac{positive-a-results}{total-a-results}$. The "-1" is for the lower bound of our probability range, and the "2*" is to scale the result to this range. Call this our *adaptive* memory storage technique:

Adaptive Memory Storage of $E_{\phi,a,r}$ in $\mathbf{Mem}[\theta]$

- $r' = r * (1 - \frac{|\phi-\theta|}{360/M})$.
- $\mathbf{Mem}[\theta]$.*total-a-results* $+= r'$.
- If $r' > 0$ Then $\mathbf{Mem}[\theta]$.*positive-a-results* $+= r'$.
- $P_a(\theta) = -1 + 2 * \frac{positive-a-results}{total-a-results}$.

For example, $E_{110,p,1}$ would set both *total-p-results* and *positive-p-results* for $\mathbf{Mem}[120]$ (and $\mathbf{Mem}[100]$) to 0.5 and consequently $P_p(120)$ (and $P_p(100)$) to 1.0. But then $E_{125,p,-1}$ would increment *total-p-results* for $\mathbf{Mem}[120]$ by .75, while leaving *positive-p-results* unchanged. Thus $P_p(120)$ becomes $-1 + 2 * \frac{.5}{1.25} = -.2$.

This method of storing to memory is effective both for time-varying concepts and for concepts involving random noise. It performs better than the basic memory storage technique described earlier because it is able to deal with conflicting examples within the range of the same memory slot.

Figure 3 demonstrates the effectiveness of adaptive memory when the defender's speed changes. In all

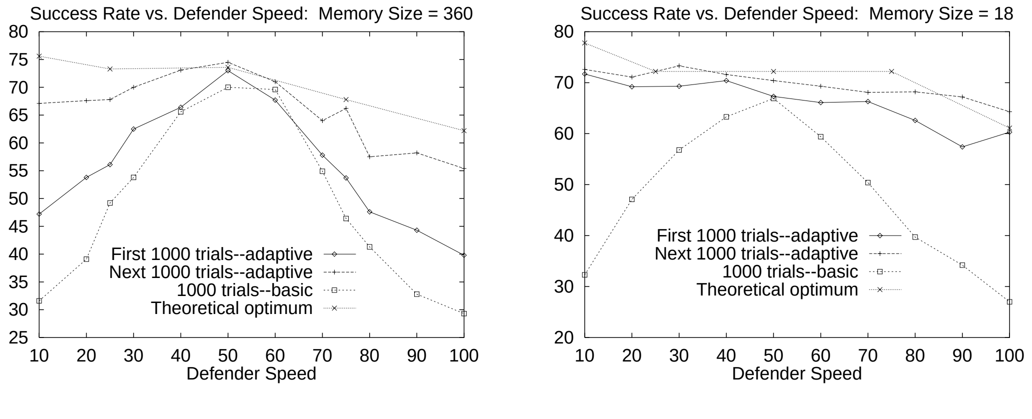

Figure 3: For all trials shown in these graphs, the agent began with a memory trained for a defender moving at constant speed 50. Adaptive memory outperforms basic memory for memories of size both 360 (left) and 18 (right). Since the basic memory does not change over time, the next 1000 trials produced the same results as the first 1000, and therefore are not plotted.

of the experiments represented in these graphs, the agent started with a memory trained by attempting a single pass and a single shot with the defender starting at each position θ for which $\mathbf{Mem}[\theta]$ is defined and moving in its circle at speed 50. We tested the agent's performance with the defender moving at various (constant) speeds.

Notice that in both graphs of Figure 3, basic memory causes performance to degrade as the defender's speed moves farther from 50. At the extremes, performance even becomes worse than random action, which leads to roughly a 40% success rate. In contrast, with adaptive memory, the agent is able to unlearn the training that no longer applies and approach optimal behavior: it *re-learns* the new setup. During the first 1000 trials the agent suffers from having practiced in a different situation (especially for the less generalized memory, $M = 360$), but then it is able to approach optimal behavior over the next 1000 trials. Remember that optimal behavior, represented in the graph, leads to roughly a 70% success rate, since at many starting positions, neither passing nor shooting is successful. As in Table 1, we can see that the smaller memory converges to P^*_a more quickly than does the larger memory.

From these results we conclude that our adaptive memory can effectively deal with time-varying concepts. It can also perform well when the defender's motion is nondeterministic, as we show next.

3.3 Coping with Noise

To model nondeterministic motion by the defender, we set the defender's speed randomly within a range. For each attempt this speed is constant, but it varies from attempt to attempt. Since the agent observes only the defender's initial position, from the point of view of the agent, the defender's motion is nondeterministic.

First, observe that this nondeterminism makes the task more difficult. The following table shows that with either type of memory scheme, the agent performs significantly worse when the defender's speed varies than when it remains constant.

	Speed 50 (2000 trials)	Speed 30-70 (4000 trials)	
Memory Size	Basic Memory	Basic Memory	Adaptive Memory
18	70.3%	60.1%	64.0%
360	73.5%	60.9%	61.2%

Even with the advantages of twice as many trials in which to learn and of an adaptive memory, the agent does not score as often when the defender's speed varies nondeterministically.

This set of experiments was designed to test the effectiveness of adaptive memory when the defender's speed was both nondeterministic and different from the speed used to train the existing memory. The memory was initialized in the same way as in Section 3.2 (for defender speed 50). We ran experiments using both types of memory in which the defender's speed varied between 10 and 50 or between 60 and 100. We compared agents with trained memories against agents with initially empty memories as shown in Figure 4.

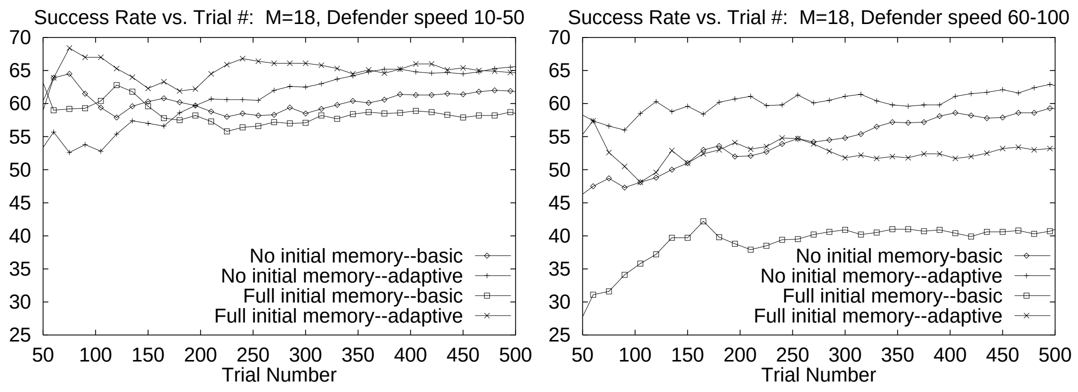

Figure 4: A comparison of the effectiveness of starting with an empty memory versus starting with a memory trained for a constant defender speed (50) different from the defender speed during testing. The performances of both adaptive and basic memories are plotted. Success rate is measured as goal percentage thus far.

Notice in both cases, the agents with full initial memories outperformed the agents with initially empty memories in the short run. The agents learning from scratch did better over time since they didn't have any training examples from when the defender was moving at a fixed speed of 50 affecting their memories;

but at first, the training examples for speed 50 were better than no training examples. Adaptive memory outperformed basic memory in both graphs.

Going back to the soccer scenario, practicing against a particular goalie is better than not practicing at all, even if the opposing goalie moves randomly and differently from the training goalie. When you would like to be successful immediately upon entering a novel setting, adaptive memory allows training in related situations to be effective without permanently reducing learning capacity.

4 Related and Future Works

This work was inspired in large part by the *Dynamo* project at the University of British Columbia. In particular, Kanazawa worked within this framework to learn when to shoot and when to pass using purely analytical methods [3]. However, since there was no learning involved, the routines had to all be hand-coded.

Christiansen has done extensive work examining the difference between analytical and learning methods [2]. His recent work relates to learning actions in a single-player, continuous-valued environment with static obstacles [16].

Aha and Salzberg use a memory-based approach to learn a continuous function without modelling the physics of the system [1]. Their task, catching a ball, differs from ours in that it has more inputs, a continuous output, and sharp nonlinearities, but they don't attempt to adapt to changes in the environment or contend with an adversary.

Sutton and Whitehead discuss the distinction between off-line and online learning [14]. As they point out, online learning has the advantage of being able to cope with changing concepts. However online techniques must be incremental so that they need not reprocess the entire training set every time a new example is added. Sutton and Whitehead distinguish between "weakly incremental" and "strictly incremental" techniques: the former require a limited amount of increased memory and computation for additional training examples, while the latter need no increase. Like their examples of STAGGER [10] and connectionist learning methods, our adaptive memory is strictly incremental. Furthermore, unlike many memory-based algorithms (as pointed out by Moore [6]) our adaptive memory is able to learn changing concepts to a certain extent. However, it also captures the notion of resiliency as discussed by Schlimmer [11]: the more training examples it has supporting a concept, the harder it is to unlearn that concept.

Previous work relating to learning changing concepts is reported in [4, 6, 9, 11]. Salganicoff's work is particularly interesting in that it introduces the notion of forgetting training examples based not on how long ago they were introduced, but instead on how closely they correlate with recent examples [9]. We will investigate using Salganicoff's ideas in our future work. Complementary to algorithms that handle changing concepts, there has been some research on adaptive algorithms that change the number of "neighbors" that are used to predict the value of an unseen instance [15]. Notice that our adaptive memory automatically captures this notion of locally adaptive algorithms by incorporating varying numbers of training examples in the different memory slots based on the number of past examples in that region.

Future work on our research agenda includes simultaneous learning of the defender and the controlling agent in an adversarial context. We will also explore learning methods with several agents where teams are guided by planning strategies. In this way we will simultaneously study cooperative and adversarial situations using reactive and deliberative reasoning.

5 Conclusion

Our experiments demonstrated that online, incremental, supervised learning can be effective at learning functions with continuous domains. We found that the degree of generalization in memory affected the speed of learning. Using a memory size appropriate to our task, we then saw that adaptive memory made it possible to learn both time-varying and nondeterministic concepts. Finally, we demonstrated that short-term performance was better when acting with a memory trained on a concept related to but different from the testing concept, than when starting from scratch. This paper reports extensive experimental results on our work towards multiple learning agents, both cooperative and adversarial, in a continuous environment.

Acknowledgements

We thank Justin Boyan for his helpful suggestions. 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] David W. Aha and Steven L. Salzberg. Learning to catch: Applying nearest neighbor algorithms to dynamic control tasks. In P. Cheeseman and R. W. Oldford, editors, *Selecting Models from Data: Artificial Intelligence and Statistics IV*. Springler-Verlag, New York, NY, 1994.

[2] A. D. Christiansen, M. T. Mason, and T. M. Mitchell. Learning reliable manipulation strategies without initial physical models. *Robotics and Autonomous Systems*, 8:7–18, 1991.

[3] Keiji Kanazawa. Sensible decisions: Toward a theory of decision-theoretic information invariants. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 973–978, 1994.

[4] A. Kuh, T. Petsche, and R.L. Rivest. Learning time-varying concepts. In *Advances in Neural Information Processing Systems 3*, pages 183–189. Morgan Kaufman, December 1991.

[5] Rod A. Barman Michael K. Sahota, Alan K. Mackworth and Stewart J. Kingdon. Real-time control of soccer-playing robots using off-board vision: the dynamite testbed. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 3690–3663, 1995.

[6] A.W. Moore. Fast, robust adaptive control by learning only forward models. In *Advances in Neural Information Processing Systems 3*. Morgan Kaufman, December 1991.

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

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

[9] Marcos Salganicoff. Density-adaptive learning and forgetting. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 276–283, 1993.

[10] J.C. Schlimmer and R.H. Granger. Incremental learning from noisy data. *Machine Learning*, 1:317–354, 1986.

[11] J.C. Schlimmer and R.H. Granger Jr. Beyond incremental processing: Tracking concept drift. In *Proceedings of the Fiffth National Conference on Artificial Intelligence*, pages 502–507. Morgan Kaufman, Philadelphia, PA, 1986.

[12] Peter Stone and Manuela Veloso. Broad learning from narrow training: A case study in robotic soccer. Technical Report CMU-CS-95-207, Computer Science Department, Carnegie Mellon University, 1995.

[13] Peter Stone and Manuela Veloso. Beating a defender in robotic soccer: Memory-based learning of a continuous function. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, Cambridge, MA, 1996. MIT press.

[14] Richard S. Sutton and Steven D. Whitehead. Online learning with random representations. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 314–321, 1993.

[15] Dietrich Wettschereck and Thomas Dietterich. Locally adaptive nearest neighbor algorithms. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 184–191, San Mateo, CA, 1994. Morgan Kaufmann.

[16] Nathaniel S. Winstead and Alan D. Christiansen. Pinball: Planning and learning in a dynamic real-time environment. In *AAAI-94 Fall Symposium on Control of the Physical World by Intelligent Agents*, pages 153–157, New Orleans, LA, November 1994.