Reinforcement Learning:

An Alternative Approach to Machine Intelligence

Capt. Leemon C. Baird, III, 1st Lt. Mance E. Harmon,

and Dr. A. Harry Klopf

Capt. Leemon C. Baird, III, 1st Lt. Mance E. Harmon, and Dr. A. Harry Klopf

Machine Intelligence Paradigms

There are many unsolved problems that computers could solve if the appropriate software existed. Flight control systems for new, nonlinear aircraft, automated manufacturing systems, and sophisticated avionics systems all involve difficult, nonlinear control problems. Many of these problems currently are unsolvable, not because current computers are too slow or have too little memory, but simply because it is too difficult to calculate what the program should do. If a computer could learn to solve the problems through trial and error, that would be of great practical value. Attempts to solve this problem are generally known as *artificial intelligence* or *machine intelligence*.

There have traditionally been two approaches, or *paradigms*, for creating useful machine intelligence. The first, the *symbolic processing* paradigm, assumes that we can tell the computer all the relevant facts in a situation, using a language of a higher level than Ada or C. The computer is then expected to logically reason out what it should do, using a large number of if-then type rules. For example, we might tell the computer all the facts about navigation and threat avoidance, then expect it to logically deduce the best route for an aircraft to fly. Expert systems are the best example of this approach.

The second approach, the *supervised learning* paradigm, doesn't assume that we know as much. We need only know a set of questions with the right answers. For example, we might not know the best way to program a computer to recognize an infrared picture of a tank, but we do have a large collection of infrared pictures, and we do know whether each picture contains a tank. The computer is expected to look at all the examples with answers and learn on its own how to recognize tanks in general. This is usually done by simulating simple neuron-like equations connected together to form networks, which then learn to recognize patterns. Standard neural networks are the best example of this approach.

Unfortunately, there are many situations where we don't know enough about the world to build an expert system, and we don't even know the correct answers that supervised learning requires. For example, in a flight control system, the question would be the set of all sensor readings at a given time, and the answer would be how the flight control surfaces should move during the next millisecond. Simple neural networks can't learn to fly the plane unless there is a set of known answers, so if we don't know how to build a controller in the first place, simple supervised learning won't help.

That is why there has been much interest recently in a different approach, the *reinforcement learning* paradigm. In this approach, the computer is simply given a goal to achieve. The computer then learns how to achieve that goal by trial and error. A reinforcement learning program works in real time with continual feedback from its environment to achieve a given goal. This real-time, closed-loop, goal-seeking behavior seems to be a crucial aspect of how brains work, and also seems to be important for creating the machine intelligence we need to solve some difficult problems. Therefore, we and others have been pursuing this form of machine intelligence and are excited about the possibility to understand large networks that are reinforcement learning controllers. These may give machine intelligence many of the capabilities that only humans have now.

Reinforcement Learning Defined

Reinforcement learning is a difficult problem because the controller may perform an action but not be told whether that action was good or bad. For example, a learning autopilot program might be given control of a simulator and told not to crash. It will have to make many decisions each second, then after acting on thousands of decisions, the aircraft might crash. What should the controller learn from this experience? Which of its many actions were responsible for the crash? It is this problem of assigning responsibility to individual actions that makes reinforcement learning difficult.

There is a solution to this problem. It is based on a field of mathematics called *dynamic programming*, and it involves just two principles. First, if an action causes something bad to happen immediately, such as crashing the plane, the controller learns not to do that action in that situation again. So whatever the controller did one millisecond before the crash, it will avoid doing in the future. But that principle doesn't help for all the earlier actions that didn't lead to immediate disaster.

The second principle is that if all the actions in a certain situation leads to bad results, that situation should be considered bad. So if the controller has experienced a certain combination of altitude and airspeed many different times, trying a different action each time, and all actions led to something bad, it will learn that this situation is bad. This is a powerful principle because it can now learn without crashing. In the future, any time it chooses an action that leads to this particular situation, it will immediately learn that that action is bad without having to wait for the crash.

By using these two principles, a reinforcement learning system can actually learn to fly a plane or control a robot or do any number of tasks. It can first learn on a simulator, then fine-tune on the real thing. All of the facts that are learned about actions and situations are typically stored using a neural network so reinforcement learning research profits from all of the accumulated experience of neural network researchers. Reinforcement learning is not a type of neural network nor is it an alternative to neural networks. Rather, it is an orthogonal approach that addresses a different, more difficult question, and the two fields can be combined to yield powerful machine-learning systems.

Advantage Updating

A reinforcement learning algorithm that we have developed, called *advantage updating* (Air Force patent pending, Baird 1994), is a particular type of reinforcement learning that is specifically designed for problems with real-time control and many actions per second. It can be applied to both optimization problems, such as autopilots, and game problems, such as dogfights. In game theory, a *game* is a problem where there are two players, and each must make decisions based on the behavior of the opponent. For example, if an anti-aircraft missile is targeted at an aircraft, the best strategy for the aircraft depends on the behavior of the missile, and the best strategy for the missile depends on the behavior of the aircraft. It is usually extremely difficult to calculate the optimal strategies for both players in a game problem, but advantage updating can be used to learn those optimal strategies. An advantage updating system can simulate multiple games with a missile and plane starting in various configurations, and can then learn the optimal actions for both the plane and the missile. After it has learned, the program can be installed in a missile, and it will play perfectly against any plane, or it can be installed in a plane, and it will play perfectly against any missile. If both the missile and the plane are controlled by the learning system, both will play well, and neither will be able to improve its performance by using any other strategy. If there is an optimal strategy, the advantage updating algorithm will always learn it, given sufficient time and memory.

We tested this algorithm by allowing it to solve a problem where a fast missile is targeted at a slower plane. The program was told only that the missile should try to hit the plane and that the plane should try to avoid the missile. This information was conveyed by using a scalar reinforcement signal that indicated how well each player was accomplishing its goal. In this case the reinforcement signal was the distance between the two players. The goal of the missile was to minimize the distance, while the goal of the plane was to maximize that same distance. The program was not told how missiles and planes work and was not given any hints about which strategies to use, so it was forced to learn all of the strategies on its own. For this particular problem, we also were able to calculate the best strategy for the missile and for the plane and so were able to compare what the program learned to the best possible answer. It was found that the advantage updating system, through learning, converged to optimal answers. Figure 1 shows the results of pitting a missile against a plane when both are following the policy that the computer learned. (In the figure, each successive diagram displays a larger geographical area, so the triangles appear smaller.) If either player were to deviate from the policy, its performance would be worse. Each of the figures show the same players and strategies but with different initial positions and velocities.

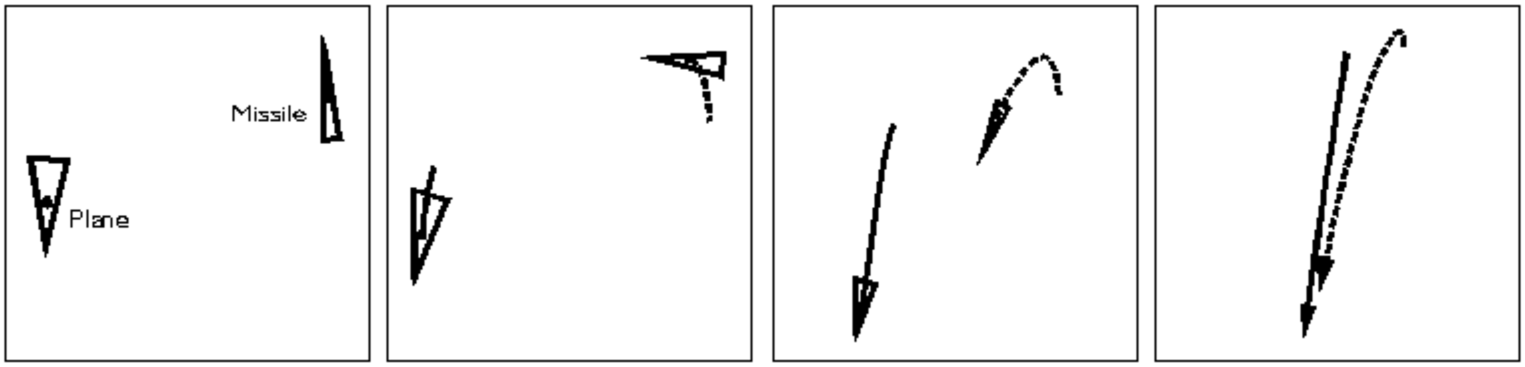

Figure 1: Progression From Initial State to Missile Impact.

The advantage updating system learned that, in some cases, it is better for a plane to turn toward the missile, decreasing the distance in the short term, to increase the distance between the two in the long term (Figure 2), a tactic sometimes used by pilots. Included in Figure 2 is a graph of distance vs. time showing the effects of the plane's learned decisions. The system also learned that a missile should sometimes lead the plane, aiming at a point in front of the plane rather than simply homing in on a heat source or radar signature (Figure 3). This demonstrates that a single advantage learning system can solve a game problem, finding the optimal actions for both players, even when it is given no initial information and must learn everything on its own. These results were presented at the Neural Information Processing Systems Conference in December 1994.

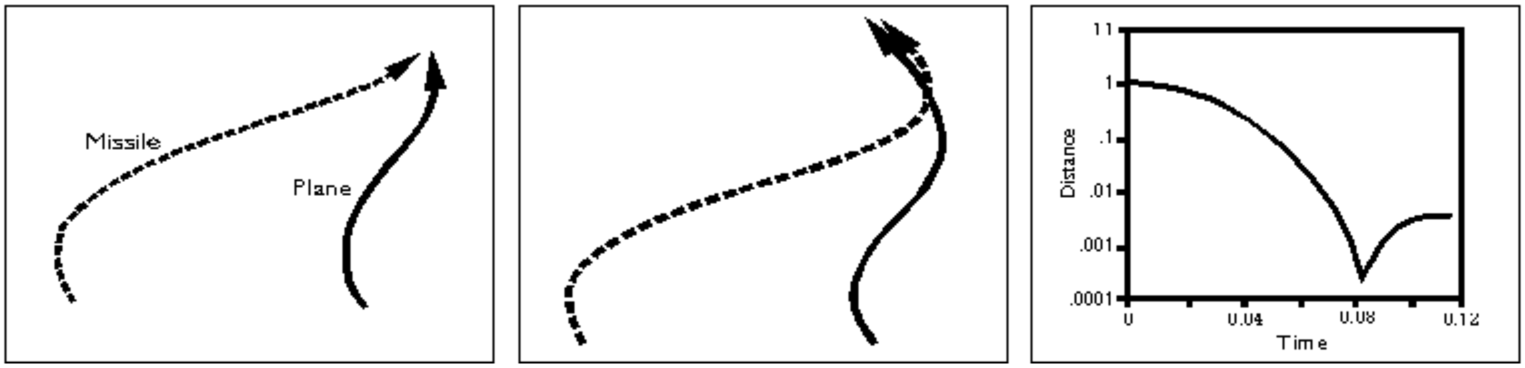

Figure 2: Plane Turning Toward The Missile In The Short Term.

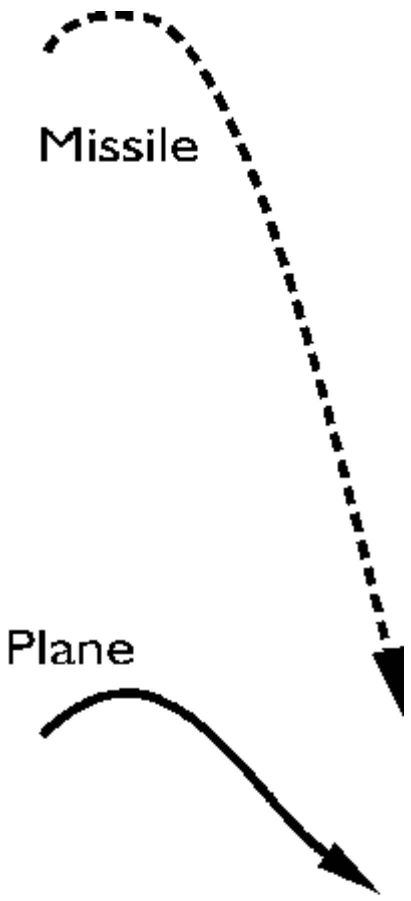

Figure 3: Missile Aims At A Point In Front Of The Plane.

Transitioning the Technology

We have worked in-house on the theoretical underpinnings of intelligent, reinforcement-learning controllers. Our work in reinforcement learning has been supported by the Air Force Office of Scientific Research, and we have also initiated external application contracts supported by Wright Laboratory. The application potential appears to be substantial for this new form of machine intelligence, so applications are currently being investigated for navigation, sensor resource management, and automatic target recognition, tracking, and pursuit.

A small business innovation research (SBIR) contract has been completed to apply advantage updating to a computer vision system. The reinforcement learning system decides how to move a motorized camera so that objects in a scene can be recognized quickly. In addition, a $1 million exploratory development contractual program has begun to apply reinforcement learning systems to avionics problems.

The field of reinforcement learning appears to have great potential for creating software and hardware that learns on its own to solve difficult problems. In the next few years, both the theory and the applications may grow exponentially, with significant impact on both the Air Force and industry.

Capt. Leemon C. Baird, III
United States Air Force Academy
HQ USAFA/DFCS
2354 Fairchild Dr. _ Suite 6K41
USAFA, CO 80840-6234
E-mail:baird@cs.usafa.af.mil

Points of Contact:
1st Lt. Mance E. Harmon and Dr. A. Harry Klopf
Wright Laboratory
WL/AAAT, Bldg. 635
2185 Avionics Circle
Wright-Patterson Air Force Base, OH 45433-7301
Voice: 513-255-7649/5800 DSN 785-7649/5800
Fax: 513-476-4302 DSN 785-4302
E-mail:klopfah@aa.wpafb.af.mil
harmonme@aa.wpafb.mil

About the Authors

Capt. Leemon Baird received a bachelor's degree in computer science from the U.S. Air Force Academy in 1989 and a master's degree in computer science from Northeastern University in 1991. He has performed reinforcement learning research at Draper Laboratory, Boston, Mass., and Wright Laboratory, Wright-Patterson Air Force Base, Ohio and is currently on the faculty of the Department of Computer Science at the Air Force Academy. More information about Baird and downloadable papers are available at **http://kirk.usafa.af.mil/~baird**.

1st Lt. Mance E. Harmon graduated from Mississippi State University with a bachelor's degree in computer science and started performing research at Wright Laboratory when he entered active duty in 1993. His interests include developing and applying reinforcement learning algorithms to problems with continuous time, state, and action sets. More information about the reinforcement home page and downloadable papers are available at **http://www.aa.wpafb.af.mil/~harmonme**.

Dr. A. Harry Klopf is the senior scientist for machine intelligence, Avionics Directorate, Wright Laboratory, Aeronautical Systems Center, Wright-Patterson Air Force Base, Ohio. Klopf leads an intramural team of scientists and engineers at Wright Laboratory, with the objective of modeling the learning mechanisms and network architectures of natural intelligence for transitioning to machine intelligence.

Klopf began his affiliation with the Air Force when he assumed a National Research Council postdoctoral associateship at the Air Force Cambridge Research Laboratories, Hanscom Air Force Base, Mass. in 1970. In the postdoctoral position and continuing to the present day, Klopf's research has focused on reinforcement learning and adaptive behavior as approaches to the analysis of natural intelligence and the synthesis of machine intelligence. Since 1973, Klopf has been with the Avionics Directorate of Wright Laboratory.

Suggested Reading

1. Baird, L.C., *Advantage Updating*, Wright Laboratory Technical Report WL-TR-93-1146 (available from the Defense Technical Information Center, Cameron Station, Alexandria, VA 22304-6145), Wright-Patterson Air Force Base, Ohio, 1993.

2. Crites, R., and A. Barto, "Improving Elevator Performance Using Reinforcement Learning," *Advances in Neural Information Processing Systems 8*, D. Touretzky, M. Mozer, and M. Haselmo, eds., MIT Press, Cambridge, Mass., 1995.

3. Harmon, M.E., L.C. Baird, and A.H. Klopf, "Advantage Updating Applied to a Differential Game," *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. Touretzky, and T. Leen, eds., MIT Press, Cambridge, Mass., 1994.

4. Klopf, A.H., "A Neuronal Model of Classical Conditioning," *Psychobiology*, 16(2), pp. 85-125, 1988.

5. Klopf, A.H., J.S. Morgan, and S.E. Weaver, "A Hierarchical Network of Control Systems That Learn: Modeling Nervous System Function During Classical and Instrumental Conditioning," *Adaptive Behavior*, 1(3), pp. 263-319, 1993.

6. Tesauro, G., "Practical Issues in Temporal Difference Learning," *Machine Learning*, 8(3/4), pp. 257-277, 1992.

7. Watkins, C, and P. Dayan, "Technical Note: Q-Learning," *Machine Learning*, 8(3/4), pp. 279-292, 1992.

8. Zhang, W., and T. Dietterich, "High-Performance Job-Shop Scheduling with a Time-Delay TD(lambda) Network," *Advances In Neural Information Processing Systems 8*, D. Touretzky, M. Mozer, and M. Haselmo, eds., MIT Press, Cambridge, Mass., 1995.