AI Magazine Volume 6 Number 3 (1985) (© AAAI)

The Emergence of Artificial Intelligence: Learning to Learn

Peter Bock

Department of Electrical Engineering and Computer Science The George Washington University, Washington, D.C. 20052

Abstract

The classical approach to the acquisition of knowledge and reason in artificial intelligence is to program the facts and rules into the machine. Unfortunately, the amount of time required to program the equivalent of human intelligence is prohibitively large. An alternative approach allows an automaton to learn to solve problems through iterative trial-and-error interaction with its environment, much as humans do. To solve a problem posed by the environment, the automaton generates a sequence or collection of responses based on its experience. The environment evaluates the effectiveness of this collection, and reports its evaluation to the automaton. The automaton modifies its strategy accordingly, and then generates a new collection of responses. This process is repeated until the automaton converges to the correct collection of responses. The principles underlying this paradigm, known as collective learning systems theory, are explained and applied to a simple game, demonstrating robust learning and dynamic adaptivity.

Foreword

As an AI researcher, I am impressed by the many recent specialized advances in the simulation of intelligence, particularly in the area of expert systems: impressed, but unsatisfied. I am still tantalized by the prospect of building a machine that exhibits general artificial intelligence—one capable of recognizing and solving a wide and ever-expanding variety of problems in diverse application domains, capable of rising to the new problem of the moment by adapting existing expertise to the solution of a new problem through analogy, of subordinating long-term goals to sets of shorter term goals that must be achieved first in order to tackle the long-term goals successfully—and even capable of creating something new and beautiful just for its own sake.

All of these capabilities are ones that we associate, in part, with human intelligence. If we accept the premise that the human brain is the seat of these capabilities, and if we accept the notion that the human brain is a finite machine, albeit extremely complex, then we arrive at the inescapable conclusion that we ought to be able to replicate such a machine with a finite number of components in a finite amount of time.

Even if we insist that the capacity of the brain is infinite (though somehow contained in a finite volume), and that its operation is magical and mystical and, therefore, unfathomable, then perhaps we may at least accept the proposition that we can build a machine that will approach our cognitive capabilities asymptotically, so that the between difference its intelligence and ours will be indistinguishable.

Many insist that even this compromise is not possible—that it is arrogant and naive to assume that "we limited humans can duplicate the infinite magic of our brains." Which is it now? Are our human abilities limited, or are they infinite? Personally, I find it rather more arrogant to assert that our intelligence capacity is infinite than to accept its finite limitations.

This topic is very controversial. When discussing artificial intelligence, people who are otherwise calm often become very emotional, and will go to absurd lengths to convince me that the endowment of machines with rational and emotional intelligence as rich and complex as ours is fundamentally impossible. When I was abroad a few years ago, I had just this kind of lively discussion with a professor of philosophy at a prestigious university. I proposed to him a classic gedankenexperiment.

Me: Suppose we build a machine that is capable of making an exact copy of an object. Now suppose we put you in this machine while you are unconscious, and make a duplicate of you. Then we take you and your duplicate and

place you in a room together. Both of you are awakened simultaneously, and presented with the task of determining which of you is the original, which of you is the copy. How would you accomplish this?

Him: Oh! That's easy. I'd know that I was the original!

Me: But your counterpart would say the same thing!

Him: Yes. But he'd be wrong.

QED. Well, everyone is entitled to his or her view. My view is that out intelligence is finite, realizable, and can be replicated in computer hardware and software.

My efforts to simulate human intelligence begin with a careful examination of the size and nature of our intelligence capacity and processes. Research in neurophysiology has revealed that the brain and central nervous system consist of about 10^{11} individual parallel processors, called neurons. Each neuron is capable of storing about 10^4 bits of information. The information capacity of the brain is thus about 10^{15} bits (Sagan, 1977). Much of this information is probably redundant, but let us use this high value as a conservative estimate of the memory requirements for a computer that can store all this information. Furthermore, for the sake of rapid response (which is certainly part of intelligence) we will require that high-speed random-access memory be used to store this vast amount of information. When can we expect to have high-speed memories of 10^{15} bits?

Examination of historical data reveals that the amount of high-speed random-access memory that may be conveniently accessed by a large computer has increased by an order of magnitude every six years. This growth is depicted in Figure 1, in which I have dubbed each six-year period a "generation."

If we can trust this simple extrapolation, in generation thirteen, (AD 2024-2030), the average high-speed memory capacity of a large computer will reach the 10^{15} bit information capacity of the human brain. Note that an order of magnitude error in the estimate of brain capacity results in an error of just six years in my prediction.

So much for the hardware. Now we have to fill this capacious memory with software. Of course, even adult brains are not filled to capacity. So we will assume that 10% of the total capacity, or 10^{14} bits, is the extent of the intelligence base of an adult human brain. How long will it take to write the programs to fill 10^{14} bits (production rules, knowledge bases, etc.)? The currently accepted rate for production of software, from conception to installation, is about one line of code per hour. Assuming, generously, that an average line of code contains approximately 60 characters, or 500 bits, we discover that the project will require 100 million person-years!!! (One of my students suggested that we put ten programmers on the project.) Now we can fiddle with these numbers all we want, cut the brain capacity estimate drastically, increase programmer productivity, use huge programming teams, etc., and we still come up with impossible times. We'll never get anywhere by trying to program human intelligence into our machine.

THE EVOLUTION OF HIGH-SPEED MEMORY CAPACITY

GENERATION	PERIOD	TECHNOLOGY	MEMORY CAPACITY (BITS PER COMPUTER)
1	1952 - 1958	Vacuum Tube	10^3
2	1958 - 1964	Transistor	10^4
3	1964 - 1970	SSI	10^5
4	1970 - 1976	MSI	10^6
5	1976 - 1982	LSI	10^7
6	1982 - 1988	VLSI	10^8
•	•	•	•
•	•	•	•
•	•	•	•
13	2024 - 2030	????????	10^{15}

Figure 1.

What other options are available to us? One is direct transfer. It goes something like this: I put electrodes all over your head, read out the contents of your brain (nondestructively, I hope), and transfer these data over a high-speed bus to our machine. Without belaboring the calculations, this task would require about twelve days. Only one problem remains: I haven't the faintest, foggiest notion of how to build such a device, let alone what principles it would employ, or even how to propose a research project to investigate its feasibility. If someone does, more power to him.

What's left? There must be another alternative, because intelligence is acquired every day. Every day babies are born, grow up, and in the process acquire a full spectrum of intelligence. How do they do it? The answer, of course, is that they learn.

If we assume that the eyes, our major source of sensory input, receive information at the rate of about 250,000 bits per second, we can fill the 10^{14} bits of our machine's memory capacity in about twenty years. Now we're getting somewhere.

Maybe what we must do is connect our machine brain to a large number of high-data-rate sensors, endow it with a comparatively simple algorithm for self-organization, provide it with a continuous and varied stream of stimuli and evaluations for its responses, and let it learn.

Of course the task is not that simple, but I believe

that the suggestion is on the right track—and perhaps the only track.

Introduction

Consistent with my professed mechanistic view of human intelligence, and my profound admiration for the ingenious and efficient design and function of the brain, I offer the following definition of artificial intelligence.

> *Artificial Intelligence is the ability of a human-made machine (an automaton) to emulate or simulate human methods for the deductive and inductive acquisition and application of knowledge and reason.*

An emulation of cognition requires building an automaton that uses the same algorithm to solve a problem as does the emulated living organism. This approach is often called brain-modeling. Emulation of the cognitive processes of members of the higher phyla of the animal kingdom requires the design and implementation of functional equivalents of neurons, which combine into networks. The biochemistry of the neuron is largely ignored, and the neuron is treated as a black box.

Simulation allows the use of a method of solution that is different from, although functionally equivalent to the method actually employed by the brain. This approach is often called problem-solving. Simulation of the complex dynamics of an arm or leg in goal-directed motion (*e.g.*, kicking a soccer ball) can be accomplished using a Jacobian transform, yet there appears to be no hard-wired algorithm for such a transform in the brain. The human brain does not have a shift-and-add multiplier in its wired-up repertoire. Yet the human brain can multiply. Some programming languages, especially as implemented on small machines, often do emulate the human brain's method of multiplication by using a look-up table.

Both the simulation and emulation of intelligence are truly artificial because neither attempts to copy the biochemical functions of the organism. Work of this kind uses biological experiments directed toward the creation and replication of living organisms in the laboratory using the same biochemical processes employed in nature. It may indeed be genetic engineers who will provide us with the components necessary to build our computer brain. In the same way that we specify the logic diagram for an integrated circuit today, we may in the future specify a useful sequence of nucleotides for a DNA molecule, growing our memory and associated logic *in vitro*. This kind of hardware already has a name: the biochip—and the information placed onto the biochip might be called wetware.

Returning once more to our definition of artificial intelligence, let us carefully distinguish between deductive and inductive processes. An automaton behaves according to the following equation:

$$S[t + \delta_t] = T \;\; S[t]$$

where $S[t]$ is the state of the automaton at time t, $S[t + \delta_t]$ is the state of the automaton at the time t, and T is the transformation that, when applied to $S[t]$, yields $S[t+\delta_t]$. In a deductive process, the transform T is known. Thus, if $S[t]$ is known (subject to limits of uncertainty, of course) we may calculate $S[t + \delta_t]$ and thus determine the progressive behavior of the automaton.

The transform may be a function that is continuous throughout the state space. For example, the formula $P[t + \delta_t] = (1 + r)$. $P[t]$ transforms an old principal, $P[t]$, to a new principal, $P[t+\delta_t]$, by the application if the transform $(l + r)$. Repeated application results in exponential growth of the principal, the familiar compound interest phenomenon.

Certainly we can envision much more complex continuous transforms, but it is doubtful whether we can postulate a single continuous transform or even a set of continuous transforms to account for and simulate the almost unfathomable, highly differentiated, nonhomogeneous, apparently erratic behavior of Homo sapiens.

Moving the transform from continuous to discrete space helps somewhat. By allowing the transform to be a matrix, we can allow our deductive automaton to leap great discontinuities in a single bound. To illustrate this point, consider the classical brain-teaser:

> You have a fox, a chicken, and a bag of grain. You come to a river that you must cross. There is a small boat available that can carry only yourself and one of your possessions across the river per trip. What is your optimal strategy for crossing the river with all of your possessions intact?

It is impossible to imagine the application of a continuous transform to this problem. However, it is easily approached using a set of discrete transforms representing the well-understood principles of classical deduction. Such problems have been successfully solved by automatons using discrete deductive problem-solving techniques (Michalski *et al.*, 1983).

However, it is fortunate that the state-space of this problem is very limited, the rules of the game simple and fixed, and the set of applicable deductive transforms well-understood. Consider a similar word problem, which I posed to some colleagues in Berlin a few years ago.

> You live in a geographical region that is completely surrounded by the enemy. The enemy's territory is in turn surrounded both by your friends and by allies of your enemy. In terms of your allies' and your enemies' economic, political, and cultural situations, what is your optimal strategy for survival.

The ability to solve this problem, in which the situation is not so simple, is obviously a much more critically needed ingredient of intelligence. However, needless to say, the application of deductive methods in this case will not yield satisfactory results. The transforms are simply not known, except perhaps in obvious and often critical phases, such as open warfare.

When the transform is not known, one may often use induction to find it. This procedure requires the knowledge of both the current state and the forthcoming state. The question immediately arises: "How can one know the forthcoming state until it happens?" One cannot, of course. The only approach is therefore to guess at the possible transform, apply it, evaluate the effectiveness of the output, modify the transform accordingly, reapply it, evaluate, and so forth, until one converges to a useful transform. This informal method of induction, when applied to problems on artificial intelligence, goes by several names: adaptive, self-organizing, or learning. The specific name for a machine that makes use of this trial-and-error approach is a learning automaton, or "LA" (Narendra and Thathachar, 1974).

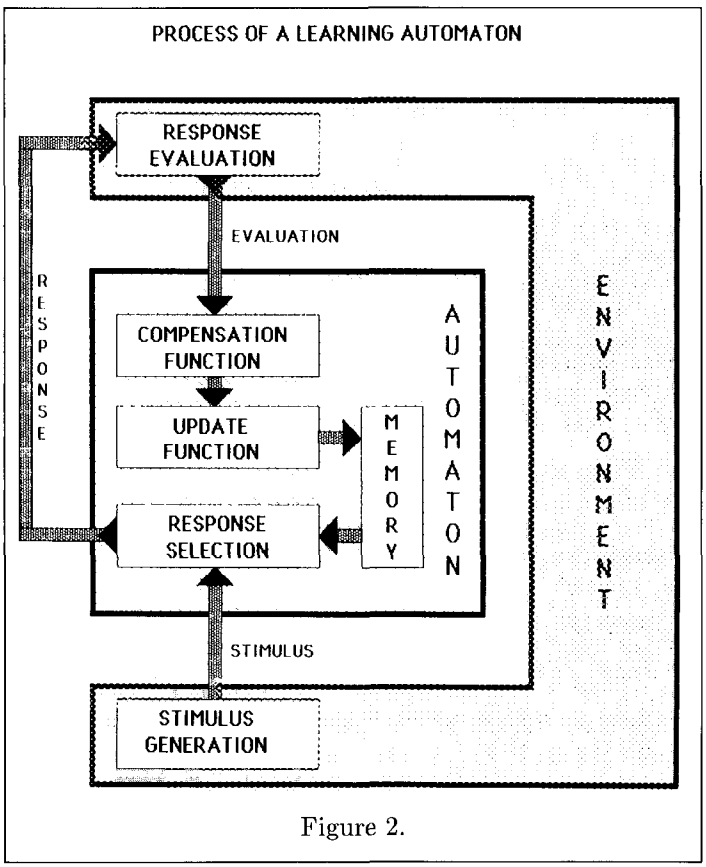


Figure 2.

Learning

The learning process is diagrammed in Figure 2. External to the learning automaton is an environment that evaluates the responses of the automaton. The environment is assumed to be consistent and infallible and to always tell the truth. In learning systems nomenclature, it thus is said to be stationary, deterministic, and correct. The environment poses the initial stimulus. The automaton then generates a trial response. The environment evaluates the response and reports its evaluation to the automaton. The automaton adjusts its strategy accordingly and generates a new trial response. This iterative process continues until the automaton converges to the correct response; that is, when the stimulus produced by the environment always (or as often as necessary) elicits the correct response form the automaton.

With this type of environment, the process is exactly equivalent to an exhaustive search of the response space until the correct response is identified. This is because every response is always evaluated by the environment as either correct or incorrect; it is, in fact, a simple process of elimination. Linear algebra provides us with the same deductive method of solution. Quite obviously, this model is not useful for complex problem solving; real-life situations generally do not allow us to evaluate each response in a complex task as it happens.

Some years ago I learned to fly. Early in my training, my instructor began the process of teaching me how to land—an essential skill indeed. One day, about 1000 meters from the end of the runway, at an altitude of about 300 meters, he turned the aircraft over to me. "Land it," he said.

I had read the books, studied the principles, and properly prepared myself. Nevertheless, this was the real thing, and I was somewhat apprehensive. At any rate, I pulled myself together, took a deep breath, and began to make my approach. To make a long story short, I managed to get the aircraft down in one piece, although via a rather unorthodox trajectory. I felt rather pleased with myself.

My instructor had said or done very little during this process except for taking control of the aircraft after the eighth bounce. When we rolled to a stop, I turned to him and asked matter-of-factly, "How was that?" He turned to me and said, equally matter-of-factly, "Terrible." And then, "That's all for today."

That was it. I received only one single word to use as an evaluation of the entire complex landing process. I had to apply this compensation collectively to the entire sequence of responses I had just generated: not one evaluation for each response; rather, one evaluation for the entire set of responses.

To be fair, I must hasten to add that my instructor did go into much greater detail during my next lesson, and I was able to apply a different and more correspondent evaluation to each of a large number of subsets of the entire response set. This debriefing was naturally much more valuable.

A TYPICAL CONTEST FOR THE GAME OF NIM

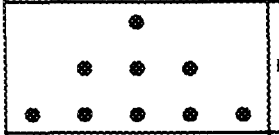
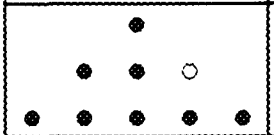
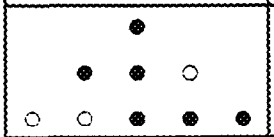
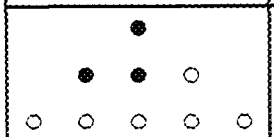
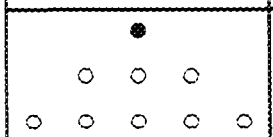
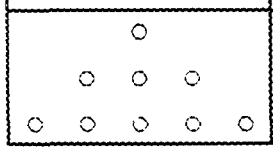
Board State	Player Responses
	Initial State of the Game
	Player A takes 1 token from the 2nd row. The 2nd row could also appear ○ ● ● or ● ○ ● since the order of tokens remaining in a single row does not matter.
	Player B takes 2 tokens from the 3rd row.
	Player A takes the 3 remaining tokens in the 3rd row. It is obvious that this is a poor move.
	Player B takes the 2 remaining tokens in the 2nd row. This is the correct move.
	Player A is forced to take the last remaining token in the first row, and thus loses the contest.

Figure 3.

I suppose it would have been most valuable if my instructor had been able to assess every component response of the process. This critique would, of course, have required him to note and evaluate every muscle contraction, resulting from every sensory input—perhaps every firing of every participating neuron. Obviously, this kind of resolution is impossible: in all complex life processes, it is unreasonable to expect the environment to provide an evaluation for every response. All evaluations are thus collective.

We can now modify the procedure employed by the LA so that an evaluation is presented to the automaton only after it has synthesized a sequence of responses, rather than after every response. I call this type of learning automaton a collective learning automaton or CLA (Bock, 1976). The theory of the interaction of the CLA with its environment and the interface between them is called Collective Learning Systems Theory (Bock *et al.*, 1985).

INITIALIZATION OF THE NIM STATE TRANSITION MATRIX

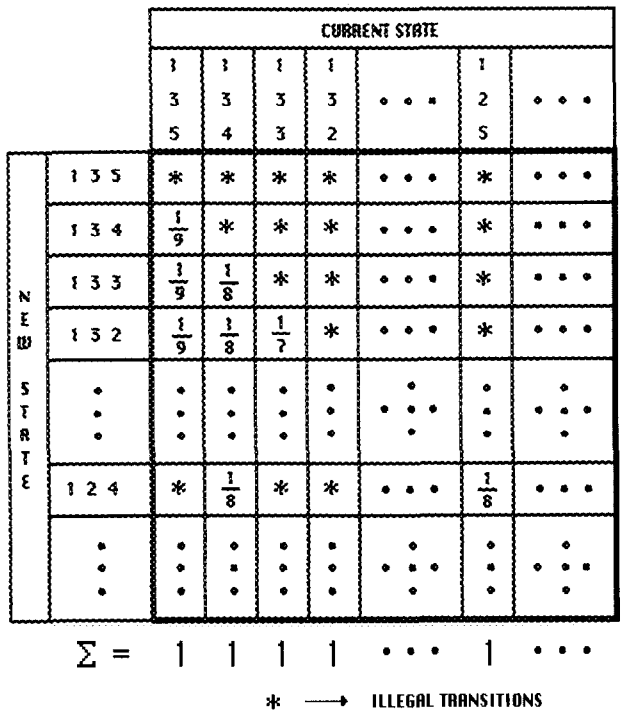


Figure 4.

An Experiment

Does collective evaluation promote learning? We suspect that the answer is yes, because collective evaluation is the rule in life, rather than the exception, and we do learn. Moreover, the concept can be approached mathematically, and it can be proved, under certain conditions, that a learning automaton will converge to a desired goal state (Narendra and Thatcher, 1974). However, I shall not dwell on the mathematics at this point. Rather, I would like to demonstrate the viability of this approach via a simple experiment.

There is an old game, sometimes called Nim, that lends itself well to demonstrate the utility of a CLA. The game begins with three rows of tokens: The first row has 1 token; the second row has 3 tokens; the third row has 5 tokens. Two players alternate turns, on each turn removing any number of tokens from any single row. The player who is forced to take the last remaining token loses. The order of the remaining tokens in any row at the end of a turn is not important—only the number of remaining tokens in each row. A typical contest is shown in Figure 3.

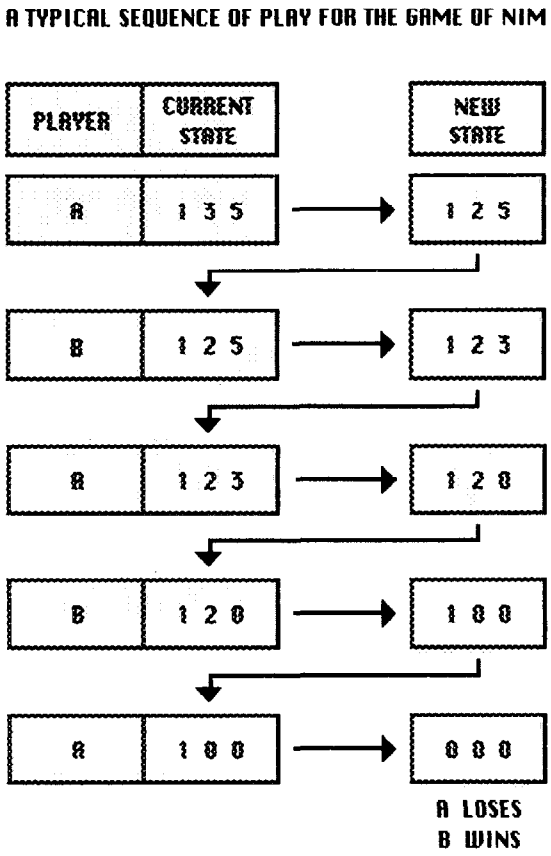


Figure 5.

Can we build a computer-based automaton that will learn how to play this game well, if not perfectly? Let us design one. First, we must count how many states the game can assume. The first row may assume two unique states (0 or 1 token); the second row may assume four states (0, 1, 2, or 3 tokens); and the third row may assume six states (0, 1, 2, 3, 4, or 5 tokens). In total, therefore, the game may assume 48 different states. Not all are unique, and some are trivial, but this is the worst case enumeration.

Now let us construct a state transition matrix, as shown in Figure 4. The numbers describing the rows and columns of the matrix correspond to states of the game, beginning with 1, 3, 5, the original state of the contest (or stimulus), and ending with 0, 0, 0, the losing (terminal) state of the contest. The column identifiers represent all possible current states of the contest, $S[t]$, and the row identifiers represent all possible successor states of the contest, $S[t+\delta_t]$, from which the player chooses its response. Each player keeps a separate matrix (memory) for its plays. Each element in the matrix represents the current probability, as estimated by the player based on its experience, that a transition from the current state to a particular state (a move) will help win the contest.

How is a matrix initialized at the beginning of a match? Some transitions are illegal. For instance, changing from state $\{1,3,5\}$ to state $\{1,2,4\}$ is illegal, since it would require removing tokens from more than one row on a single turn. Such illegal transitions are represented with asterisks in Figure 4. All other elements are initialized to be equally likely. For instance, for state $\{1,3,5\}$ there are only nine possible successor states, so each is initialized to 1/9; for state $\{1,3,4\}$ there are only eight possible successor states, so each is initialized to 1/8; and so forth.

CLA A (player A) begins by choosing a legal successor state based on the current (initial) state $\{1,3,5\}$. All successor states are equally likely at this point. Suppose it chooses $\{1,2,5\}$. Now CLA B (player B), using its transition matrix, chooses a legal successor state to the current state left by CLA A. All are still equally likely. Suppose it chooses $\{1,2,3\}$. It is now CLA A's turn again, and the contest proceeds in a similar manner until one player is forced to take the single remaining token, and so loses. Figure 5 shows the sequence of play corresponding to the contest depicted in Figure 3.

CLA A takes the last token and loses; CLA B wins. Now CLA B accesses its transition matrix and rewards each specific transition it made, by raising the probability of that transition at the expense of all other possible transition probabilities in that column of the matrix. Obviously, CLA B rewards the transitions $\{1,2,5\}$ to $\{1,2,3\}$, and $\{1,2,0\}$ to $\{1,0,0\}$. Just because CLA B won does not imply that these transitions represent a perfect strategy; after all, CLA A was also playing randomly during this initial contest. For this reason the probabilities in CLA B's transition matrix are raised only a bit, not set to unity. It obviously takes many games to converge to the optimal strategy.

CLA A similarly lowers the probabilities of all transitions that it used in its loss: $\{1,3,5\}$ to $\{1,2,5\}$, $\{1,2,3\}$ to $\{1,2,0\}$, and $\{1,0,0\}$ to $\{0,0,0\}$. In the last move, of course, it had no alternative.

Now the players begin a new contest using the updated matrices. No longer are all transitions equally likely, and the players' choices are biased by their experience. On each turn, each player now chooses the action with the highest probability of success. The contest is completed, the winner and the loser are determined, and once again the players update their transition matrices accordingly. This process repeats itself many times, until an entire match of games has been completed. Under the proper conditions, less than one hundred games is usually sufficient to cause one or the other player to converge to the optimal playing

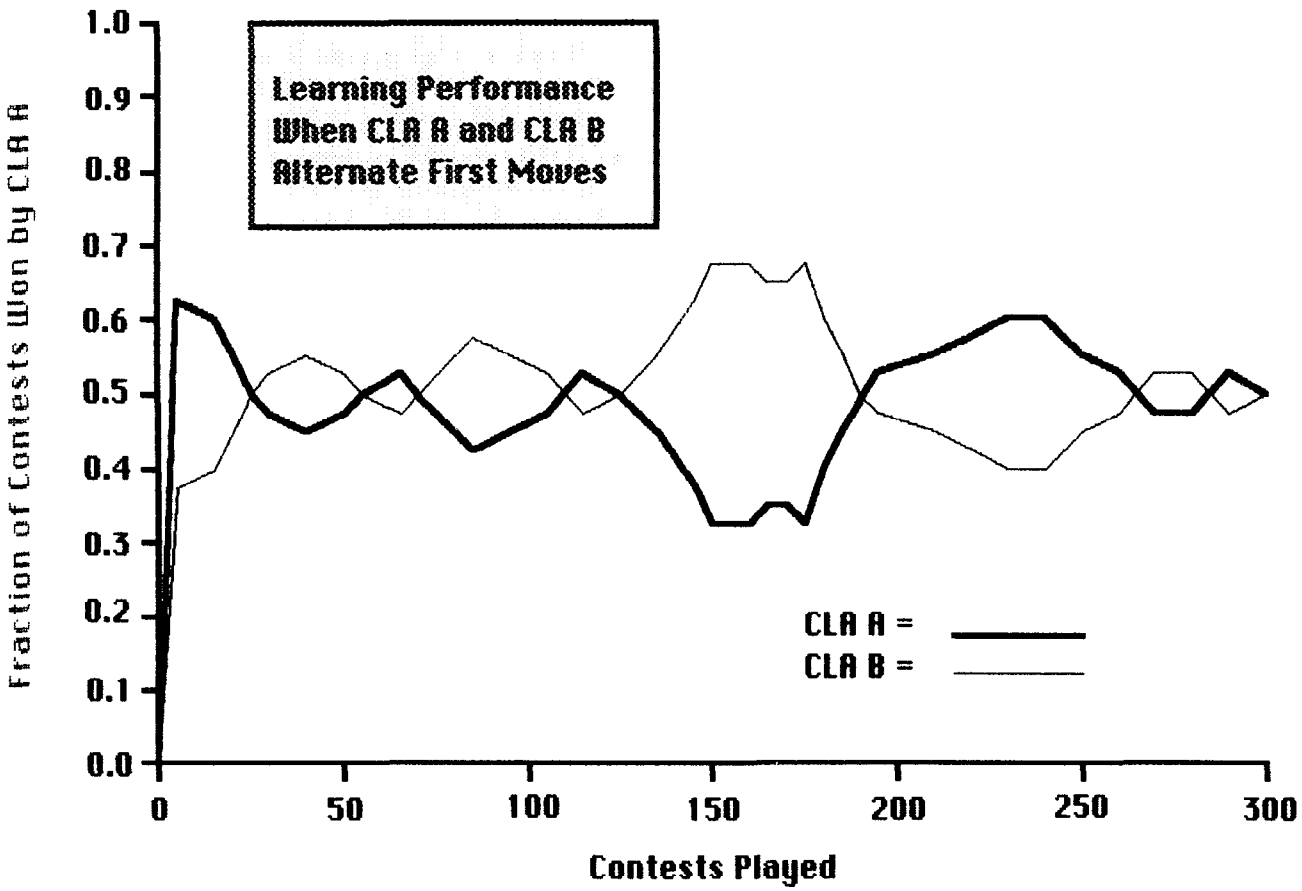


Figure 6.

strategy, if convergence is, in fact, possible.

Results

Nim is a zero-sum game, which implies, among other things, that the outcome is either a win or a loss, never a draw. Moreover, there exists an exact deterministic algorithm for the solution of the game. Therefore, the outcome of a contest by two perfect players using this algorithm must be a function of who goes first.

Given this, it is interesting to pit a learning player (a CLA) against a perfect player, allowing the CLA the privilege of always going first to measure the CLA's ability to learn the game strategy while playing an expert. The results of a 300-contest match show that the CLA loses every contest until the 65th contest, and thereafter wins them all. It is at this point, obviously, that the CLA has assembled all the elements of the perfect strategy, and subsequently always capitalizes on the definitive advantage of always going first.

Now let us initialize two learning players, CLA A and CLA B, and pit them against each other, alternating who goes first for each contest. Figure 6 shows the fraction of games won by CLA A and CLA B over a 300-contest match. As might be expected, their learning is symmetric about a line that represents winning half the time. However there are interesting surges, declines and plateaus in their learning curves. Examination of their memories (state transition matrices) reveals that each surge is a sudden insight into a piece of overall strategy, allowing the player experiencing the insight to leap ahead of the other player, but only temporarily. Soon the other player learns the same sub-strategy, and the contest evens out

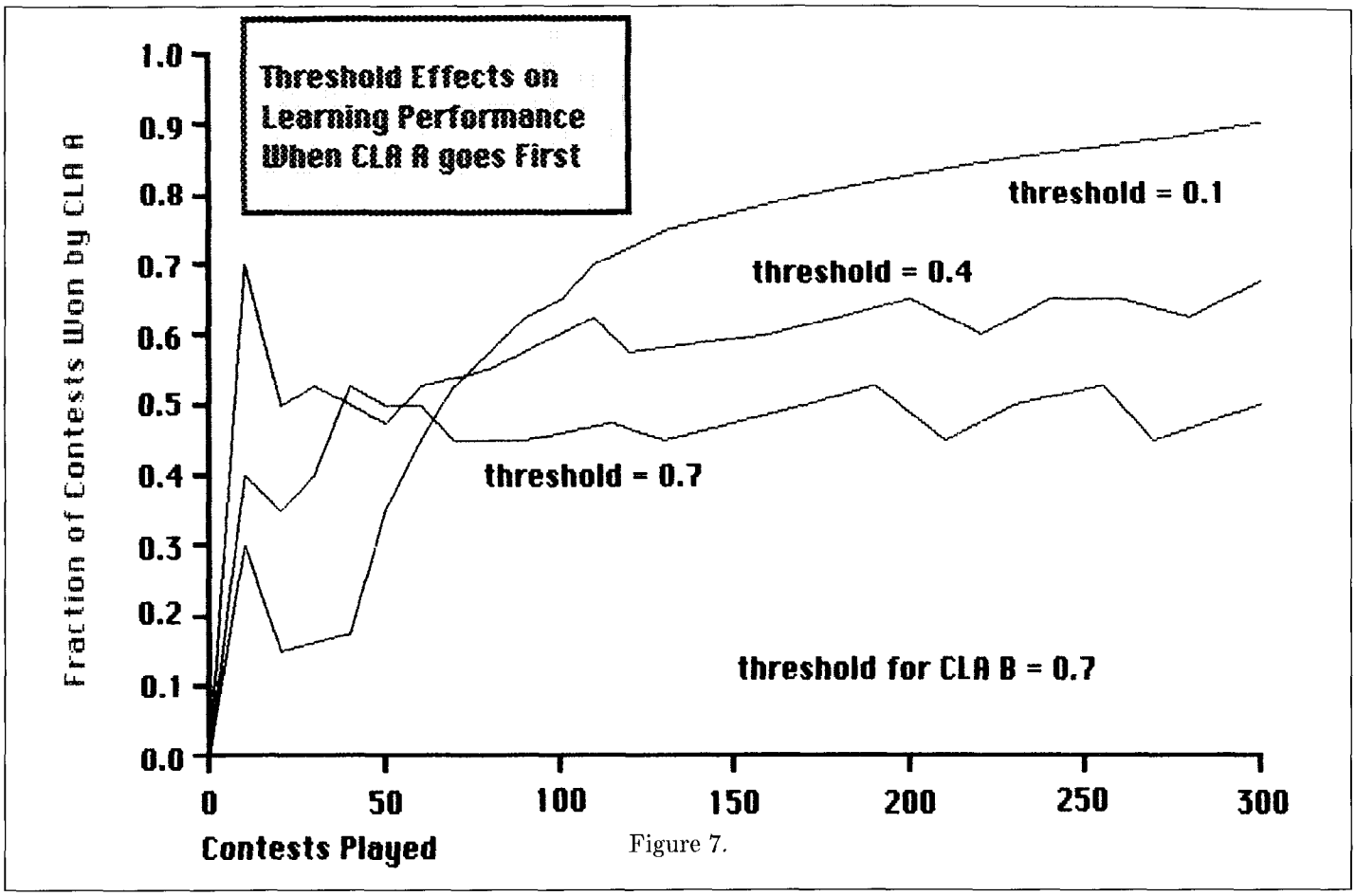


Figure 7.

again. Eventually, after 300 contests, each knows the perfect strategy and uses it to assure a win whenever it goes first, succumbing inevitably to defeat whenever it goes second.

Now let us reinitialize our two CLAs and give both learning thresholds. A learning threshold is the probability value below which the successor state with the highest transition probability will not be selected; instead, the CLA makes a completely random choice. The purpose of this strategy is to discourage the players (for various levels of discouragement) from choosing the move with the highest probability, when that probability is itself only marginally higher (for different margins) than those of all other legal moves.

For instance, if a player is using a threshold of .07, and the highest transition probability is only 0.4, the player will select its successor state randomly from among all legal successor states. On the other hand , if the highest transition probability is 0.75, the player will select the successor state corresponding to it. This policy encourages the player to range about a bit, experimenting with all legal moves until one move emerges as obviously better.

It is also interesting to vary which player goes first in a contest: CLA A, the winner of the previous contest, or the loser of the previous contest. Figures 7, 8, and 9 show the performance of CLA A over nine 300-contest matches, varying which CLA goes first, and the learning threshold employed by CLA A (0.1, 0.4, and 0.7). The threshold for CLA B is held constant at 0.7. The graphs of the results for CLA B would simply be the complement, or mirror image, of those for CLA A.

Figure 7 shows the results of three independent matches in which CLA A always goes first, using three different thresholds. The lowest threshold (0.1) yields the best long-term results by far, although early performance is poor. Apparently the immense advantage of always going first, and a low threshold to boot, encourages arrogant, high-powered play resulting in poor early performance but, eventually, total domination. When the threshold is increased to 0.4 early performance is better. However, the overall behavior of CLA A is necessarily more conservative, and CLA B has time to learn the proper defense and hold CLA A to more moderate gains. Nevertheless, CLA B must contend with being the long-term loser, the inevitable fate of never going first. Finally when the threshold is raised to 0.7, after 300 games the reactionary timidity of CLA A still prevents it from learning the perfect strategy and it cannot make use of the advantage of always going first.

In Figure 8, when the advantage of always going first

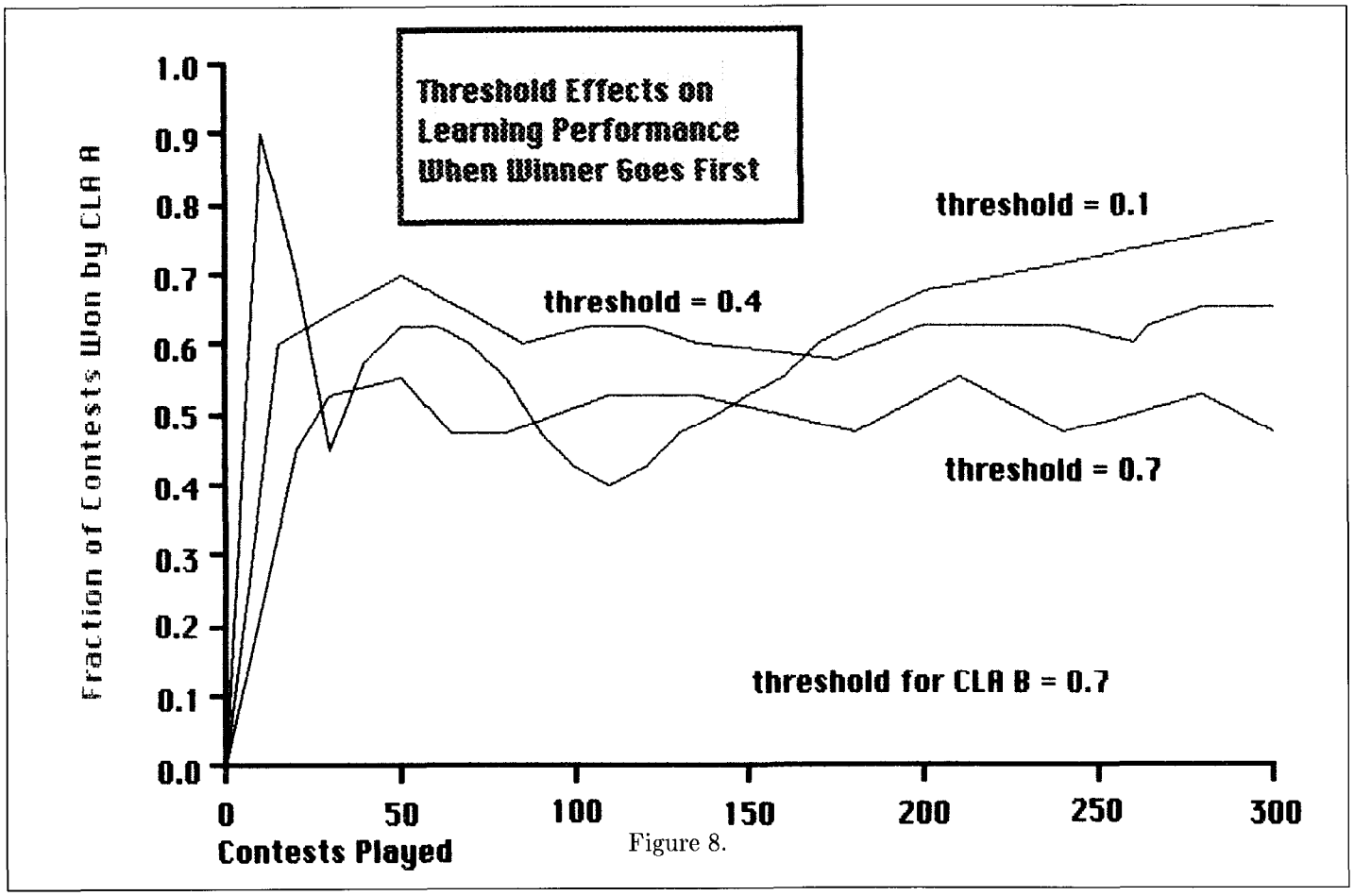


Figure 8.

is removed, and CLA A must win to go first, the lowest threshold (0.1) still turns out to be the best, although performance is not nearly as good as in the previous case. If we calculate the area under each of these curves (a measure of the long-term payoff), the conservative policy (threshold = 0.4) is the most successful.

In Figure 9, the loser always goes first. This variation has an odd effect. In order to assure a win by going first, one must lose. Successful learning relegates a player to the unfortunate status of never being able to take advantage of its knowledge by going first and winning. Both players struggle valiantly to win, and upon doing so, are "rewarded" with the disadvantage of having to go second in the next contest. The graphs reflect the results of this vicious circle. Our arrogant CLA A (threshold = 0.1) storms into the fray hoping for a lucky break. It allows itself to be drawn inexorably into incorrect absorbing states and becomes totally neurotic, desperately learning to lose, just so that it can go first. Meanwhile, the reactionary policy of CLA B (threshold = 0.7) pays off. Calmly watching the frantic demise of CLA A, CLA B remains content to be a consistent, unchanging, mediocre player against a totally inept opponent. Our conservative CLA A (threshold = 0.4) handles the inevitability of the paradox better, but still cannot do well enough to counter the reactionary policy of CLA B. Our reactionary CLA A (threshold = 0.7), like CLA B, seems to catch on to the insoluble paradox early on and contents itself with trading contests with CLA B—winning one contest to learn something, losing the next to go first.

Conclusions

Experiments of this sort are endless in their parametric variation. Bounding the effective operational domains of a collective learning automaton has proven over the years to be an enormous and elusive task. Only recently have we defined an orderly and comprehensive nomenclature, symbology. and set theoretic for collective learning systems (Weingard *et al.*, 1985), as well as a set of performance metrics and behavioral norms (Bock *et al.*, 1985). It is difficult to draw generalizations about learning from my research with CLAs, but I timorously offer a few tentative hypotheses that seem to be valid in many cases:

(1) Collective learning is a broadband phenomenon. It is typically not necessary to fine-tune the learning parameters just to achieve learning. Each parameter has a wide dynamic range that will promote learning. Optimal learning, on the other hand, requires careful adjustment of all parameters.

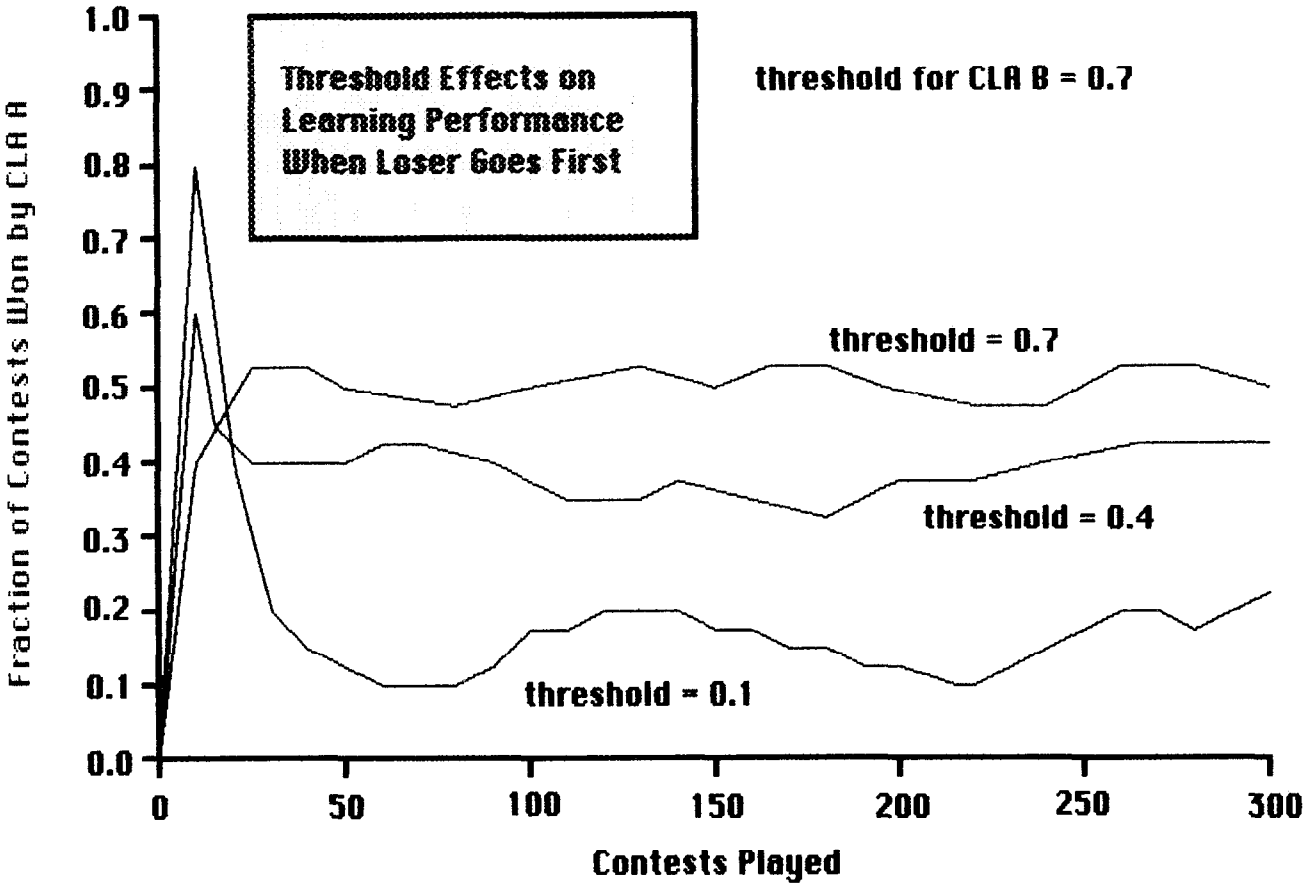


Figure 9.

(2) Keep collection lengths as short as possible, so as to dilute learning information as little as possible.

(3) Any policy of reward and punishment is more effective if it incorporates the learner's expectation of himself at the time.

(4) Make the range of the evaluation of the environment as large as possible, varying from excellent to poor with as many intermediate values as possible.

(5) Punishment should be used as a last resort, and in very small measures. Reward is a much stronger learning motivation.

The illustration of collective learning system theory for the Nim game is hardly an example of general artificial intelligence. The CLA can play only this particular game. However, it is quite simple to change the rules of the game, and have it adapt quickly to the new required strategies. By the same token, its stimulus domain and response range can be enlarged, enabling it to accomplish complex tasks. We are currently studying methods by which these modifications themselves could be learned and subsumed by an evolving CLA or set of CLAs. Successful research (Sanchez-Aguilar, 1982) has already demonstrated that a hierarchy of CLAs can be generated that recursively decomposes tasks, solving the simpler tasks first, returning eventually to solve the tasks at the highest level of the task hierarchy. Additional research is being conducted to define a genetic system by which CLAs reproduce, mutate, and die, resulting in the evolution of more complex CLAs with new and improved capabilities (Kada and Bock, 1985).

Recall for a moment my chart predicting the evolution of computer memory capacity up to generation thirteen.

What do you suppose is in store for us in generation fourteen?

References

Bock, P. (1976) *Observation of the properties of a collective stochastic learning automaton* Proceedings of the International Information Sciences Symposium.

Bock, P (1985) *Proposed performance metrics and behavioral norms for collective learning automata* International Symposium on New Directions in Computing.

Bock, P , Weingard, F., & White, J. (1985) *A generalized structure and nomenclature for collective learning systems* International Symposium on New Directions in Computing

Kada, H , & Bock, P (to be published) *A reproduction scheme for collective learning automata*

Michalski, R., Carbonell, J., & Mitchell, T. (1983) *Machine learning: an artificial intelligence approach* Palo Alto, CA: Tioga Publishing Company.

Narendra, K., & Thathachar, M. (1974) *Learning automata—a survey.* IEEE Transactions on Systems, Man, and Cybernetics SMC-4: 4

Sagan, C (1977) *The Dragons of Eden* New York: Random House.

Sanchez-Aguilar, A. (1982) HCLSA—Hierarchical collective learning stochastic automata. Ph.D dissertation, The George Washington University