Machine Learning for Prediction and Control

Gregory Galperin & Paul Viola

The Problem: The goal of this project is to investigate machine learning techniques for enabling computers to control complex and stochastic systems and predict the outcomes of such systems. More specifically, we are developing algorithms tailored to a parallel architecture which can learn from experience to perform that control task.

The first example of such a system we are investigating is learning to play the game of backgammon. Backgammon is a two-player board game in which players alternate moves and roll dice (the random element) to constrain which moves are allowable. It is far too difficult to solve exhaustively, and traditional tree-search based techniques see little success in this domain because of both its complexity and randomness.

Motivation: The development of a system which is able to automatically learn to perform a complex control task well both is of significant interest to the field of Artificial Intelligence and is of considerable practical value. Traditionally, such control systems have been hand-crafted attempts to capture an expert human's intuitions about the control task; they require tedious and extensive human effort, and results are mixed, often poor. In contrast, a successful learning machine would be able to learn such a task without an expert human even existing, and requires only computational effort; in its most successful domains, such automatic techniques have performed significantly better than the best hand-crafted efforts.

Perfect information board games are excellent arenas for testing new learning techniques. Clearly, the task is extremely difficult as a whole, and the decision task is "realistically complex:" the space of possible states is immense (10^{24} states). The environment is simple and the rules (state transitions and rewards) are well-defined, allowing for effective simulation. Finally, performance is relatively easy to measure; humans intuitively understand these domains, and thus can have insight into the computer's progress, oversights, innovations, and representation.

Backgammon is a particularly relevant game to study because of its random element; unlike completely deterministic games such as chess, in which a player can know exactly what state they are in and can cause the system to do exactly as they decide, most applications pertaining to the real world involve elements which can neither be detected nor controlled and are thus effectively random.

Previous Work: Techniques such as reinforcement learning of neural networks have been very successful in allowing a computer to learn to perform control tasks; however, their performance is limited by available serial computational power because the current algorithms do not parallelize well. [3]

The foundation for the backgammon project in this research was laid by Dr. Gerald Tesauro, a collaborator in this project, first with Neurogammon in 1989, a neural network trained on human expert move preferences. Two years later Tesauro created TD-Gammon, a neural network which learns to play from experience gained in games played against itself. TD-Gammon quickly became the strongest computer backgammon player and could beat all but a handful of the world's best humans. [2]

Approach: We have designed a Monte-Carlo simulation technique in which control decisions are made in real time by choosing the action that produces the best outcome statistics in the simulation. This procedure leads to a substantial improvement over the neural network's original control decision. [1]

We use Monte-Carlo search to estimate $V_P(x, a)$, the expected value of performing action a in state x and subsequently executing policy P in all successor states. Here, P is some given arbitrary policy, as defined by a "base controller" (we do not care how P is defined or was derived; we only need access to its policy decisions). In the Monte-Carlo search, many simulated trajectories starting from (x, a) are generated following P, and the expected long-term reward is estimated by averaging the results from each of the trajectories. (Note that Monte-Carlo sampling is needed only for non-deterministic tasks, because in a deterministic task, only one trajectory starting from (x, a) would need to be examined.) Having estimated $V_P(x, a)$, the improved policy P' at state x is defined to be the action which produced the best estimated value in the Monte-Carlo simulation, i.e., $P'(x) = \arg\max_a V_P(x, a)$.

Network	Base player	Monte-Carlo player	Monte-Carlo CPU
Lin-1	-0.52 ppg	-0.01 ppg	5 sec/move
Lin-2	-0.65 ppg	-0.02 ppg	5 sec/move
Lin-3	-0.32 ppg	+0.04 ppg	10 sec/move

Table 1: Performance of three simple linear evaluators, for both initial base players and corresponding Monte-Carlo players. Performance is measured in terms of expected points per game (ppg) vs. TD-Gammon 2.1 1-ply. Positive numbers indicate that the player here is better than TD-Gammon. Base player stats are the results of 30K trials (std. dev. about .005), and Monte-Carlo stats are the results of 5K trials (std. dev. about .02). CPU times are for the Monte-Carlo player running on 32 SP1 nodes.

We can see in Table 1 that the Monte-Carlo technique produces dramatic improvement in playing ability for these weak initial players. As base players, Lin-1 should be regarded as a bad intermediate player, while Lin-2 is substantially worse and is probably about equal to a human beginner. Both of these networks get trounced by TD-Gammon, which on its 1-ply level plays at strong advanced level. Yet the resulting Monte-Carlo players from these networks appear to play about equal to TD-Gammon 1-ply. Lin-3 is a significantly stronger player, and the resulting Monte-Carlo player appears to be clearly better than TD-Gammon 1-ply. It is estimated to be about equivalent to TD-Gammon on its 2-ply level, which plays at a strong expert level.

Difficulty: TD-Gammon represented a major advance in the state of the art in learning a control policy. Unfortunately, the amount of computation required grows drastically as you increase the size of the neural network to gain better results, and a plateau of how much a single processor could achieve was soon reached. The logical next step would be to use multiple processors, but the reinforcement learning process unfortunately does not parallelize well at all. The challenge is to develop new algorithms which exploit the power available in massively parallel machines by tailoring our approach to match the architecture of the machine.

Impact: This technique proposes a novel approach to learning, combining features of on-line and off-line methods to achieve considerable performance in the task of learning a backgammon value function in a process that exploits the processing power of parallel supercomputers. We have already created the world's best computer backgammon player using our techniques; we expect that our system is also currently better than the best humans, and we hope to demonstrate this within a year. Further, this success is hoped to transfer to other domains.

This project is breaking new ground in the field of machine learning and neural networks. The techniques we are developing will make it possible for computers to learn larger problems and to learn a given problem with greater accuracy.

Future work: On the theoretical side, we hope to complete a proof that the techniques we have developed will guarantee an improved control policy. Practically, these techniques should also have broad applicability to many other classes of real-world problems in which the environment can be simulated, such as manufacturing process control, robotic control, combinatorial optimization problems like job-shop scheduling (delivery truck scheduling, aircraft maintenance, etc.), and simulated economies and financial markets. [4]

Research Support: "Machine Learning for Prediction and Control" funded by IBM under contract number 49140046 administered by IBM T.J. Watson Research Center.

References:

[1] G. Tesauro and G. Galperin, "On-Line Policy Improvement using Monte-Carlo Search." In: M. Mozer et al., eds., Advances in Neural Information Processing Systems 9, MIT Press (1997).

[2] G. Tesauro, "Temporal difference learning and TD-Gammon." *Comm. of the ACM*, **38:3**, 58-67 (1995).

[3] R. S. Sutton, "Learning to predict by the methods of temporal differences." *Machine Learning* **3**, 9-44 (1988).

[4] W. Zhang and T. G. Dietterich, "High-performance job-shop scheduling with a time-delay TD(λ) network." In: D. Touretzky et al., eds., Advances in Neural Information Processing Systems 8, MIT Press (1996).