Successive Refinement of Empirical Models

Ralph E. Gonzalez
Department of Mathematical Sciences
Rutgers University
Camden, NJ 08102, USA
(rgonzal@gandalf.rutgers.edu)

Abstract

An approach for automatic control using on-line learning is presented. The controller's model for the system represents the effects of control actions in various (hyperspherical) regions of state space. The partition produced by these regions is initially coarse, limiting optimization of the corresponding control actions. Regions contract periodically to enable optimization to progress further. New regions and associated optimizing elements are generated to maintain coverage of state space, with characteristics drawn from neighboring regions. During on-line operation the resulting state space partition undergoes successive refinement, with "generalization" occurring over successively smaller areas. The system model effectively grows and becomes more accurate with time.

1 Introduction

An on-line learning controller maintains a model of its environment (the action model), from which control actions are determined. The model is refined during nonsupervised operation to improve control with respect to an index of performance. A *performance-adaptive* controller [Saridis, 1979] learns by direct reduction of its uncertainties, rather than explicit identification of the environmental parameters. That is, the controller's understanding of its environment is represented by its control decisions, and is constructed empirically.

An example is the *lookup-table* controller [Waltz and Fu, 1965]. The space of system states is partitioned, and each "control situation" (partition element) is associated with an independent adaptive element. Each element attempts to determine the control action appropriate within its region of state space; for example, using a probability-reinforcement scheme. The system model is a table relating each region to the response which is most likely optimal.

This paper extends the lookup-table approach by allowing regions to contract and to recursively spawn smaller regions. The resulting state space partition undergoes successive refinement during the on-line training period, and the size of the lookup-table increases accordingly.

This approach is implemented in a procedural framework, where each region is associated with an independent adaptive element whose accuracy is limited by the current region size. Beginning with large regions encourages generalization and speeds initial learning. As the regions contract, the associated adaptive elements may obtain arbitrary accuracy. Learning is accelerated in those areas of state space which arise most frequently, and the range of generalization is successively reduced. Application to the optimal control of a simulated robotic manipulator is discussed.

2 Learning Control

An automatic means of controlling a complex system is desired. The relevant aspects of the system comprise the system's *state*, denoted by the vector $\mathbf{x}$. If the dynamics of the system are unknown or are difficult to analyze, the controller may empirically learn the appropriate control vector $\mathbf{u}$ for each state, according to some available index of performance (IP) $f(\mathbf{x},\mathbf{u})$. The controller produces a model mapping the state space X onto the control actions $\mathbf{u}^*(\mathbf{x})$ which optimize the IP.

Of particular interest is the problem of on-line learning, where the controller must construct a system model while controlling the system. In this case it is desirable for the controller to be initialized with or to quickly learn control actions which maintain stable operation, and to continue to improve upon these actions over time. A robust learning algorithm is required, since the selection of states is determined by the dynamics of the system rather than by a supervisory algorithm.

2.1 State Space Partitioning

If the state space X is discrete, the learning controller may approximate $\mathbf{u}^*(\mathbf{x})$ for each state $\mathbf{x} \in X$. If the cardinality of the set of states is very large or if X is continous, then computing $\mathbf{u}^*(\mathbf{x})$ for all states may require excessive (or infinite) time. In this case, the controller may partition state space into discrete regions $\{r_i: i=1,2,...n\}$. The learning algorithm associates a single suboptimal control action $\mathbf{u}_i$ with each region, producing a lookup-table.

If the state space partition is very coarse (n is small), then the effectiveness of $\mathbf{u}_i$ may vary greatly for different states within the region r_i. On the other hand, if the partition is very fine then the overall learning time will remain large. Previous researchers have applied techniques to improve the effectiveness of a coarse partition. Waltz and Fu [1965] suggested that regions along the switching boundary of a binary controller may be partitioned once more (based on the failure of the control actions associated with these regions to converge during the training period). Albus' neural model CMAC [1975] used overlapping coarse partitions to give the effect of a finer partition while achieving greater memory efficiency. Rosen *et al.* [1992] allowed region boundaries to migrate to conform to the control surface of the system, which also resulted in a controller which adapted well to changing system dynamics. These techniques do not fully overcome the accuracy limitation which ultimately characterizes any fixed-size system model.

2.2 Contractible Adaptive Elements

The system model described here consists of a set $C=\{c_i: i=1,2,...n(t)\}$ of Contractible Adaptive Elements (CAEs), whose cardinality n is a nondecreasing function of time. Each CAE c_i associates a single suboptimal control action $\mathbf{u}_i$ with a hyperspherical region r_i of the state space X. The element c_i becomes active when the state $\mathbf{x}(t)$ moves into r_i (Figure 1a). While active, the CAE searches for an optimal control action $\mathbf{u}^*(\mathbf{x})$, according to some index of performance $f(\mathbf{x},\mathbf{u})$. Since $\mathbf{u}^*(\mathbf{x})$ varies over $\mathbf{x}\in r_i$, the search may fail to converge. Under conditions of continuity, a bound on this "noise" can be obtained (generally based on the size of the region). This enables the establishment of an accuracy threshold δ_i beyond which the search becomes ineffective.

The active CAE c_i searches until its accuracy threshold δ_i is reached. At this point, the CAE reduces the size of its region r_i (Figure 1b) and reduces δ_i. As the controller is employed on-line, each CAE's region may contract repeatedly to permit arbitrary search accuracy.

When a region contracts, it is possible for a state to arise nearby which falls outside of the boundary of all existing CAEs in the set C (Figure 1c). A new CAE is generated, whose region may be centered at the current state location (Figure 1d). The size of the new region and the starting value for the search process may be "generalized" from neighboring CAEs, with consideration of the distance to the center of their associated regions. As more nearby states arise during on-line operation, the contraction of a CAE results in repopulation of the vacated area of state space with smaller CAEs. These second-generation CAEs will themselves eventually contract, leading to still-smaller third-generation CAEs. The range of generalization contracts with each generation.

It is possible for a state to fall within the boundaries of two or more hyperspherical CAEs. Only one may be activated; e.g., by consideration of the distance from the current state to the center of the respective regions.

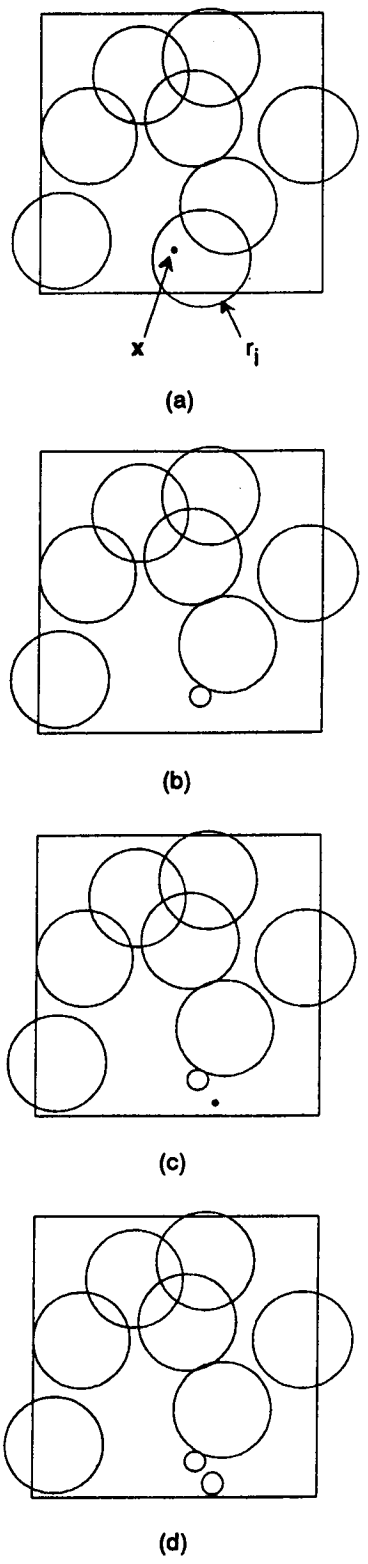

Figure 1. CAE regions on two-dimensional state space; (a) at beginning of control interval, (b) after r_i contracts, (c) at beginning of next control interval, (d) after creation of new region.

2.2.1 Performance Features

Preliminary learning is accelerated by the use of a coarse initial partition whose individual elements enjoy frequent activation. Under certain conditions, it can be proven that the successive refinement approach achieves the accuracy possible with a fixed partition in less time [Gonzalez, 1989]. The CAE approach is also more flexible than that involving a fixed partition, since in many cases the desired partition resolution is not known beforehand.

During on-line control, the state space partition defined by the set C generally becomes more refined in those areas of state space which arise most frequently. This provides memory efficiency and accelerates learning in these areas.

The optimal control action may vary more quickly over some areas of state space than others. For example, there may be a discontinuity in the value of the optimum at some surface of state space. If this surface passes through a CAE's region, then that CAE's search process will fail to converge. The region should then contract, allowing the space it vacated to be repopulated with smaller CAEs. If this occurs successively, the discontinuity will be contained in regions whose combined volume can be made arbitrarily small. Thus the likelyhood of a state arising in one of these critical regions will be equally small. On the other hand, if there is very little variation in the optimum over a large region, it is possible for the associated CAEs' search processes to use a finer accuracy threshold without contracting. This allows more rapid learning than would be the case with smaller CAEs.

2.2.2 Search Techniques

Assuming state space is continuous and the controller is in on-line operation, a particular state x will generally arise only once. Therefore it is not possible to accurately estimate the gradient $\nabla f(\mathbf{x},\cdot)$ representing the change in performance produced by a change in control action for a given state. This prevents the use of pure steepest-descent based methods when searching for control actions. If the range of control actions is itself discrete, a probability-reinforcement scheme [Waltz and Fu, 1965] may be used in place of search. If control space is continuous, direct-search optimization is used.

Direct-search methods take many forms, and generally utilize implicit estimates for the performance gradient. In our implementation, each CAE maintains an independent direct search process. Its parameters include the current suboptimal control action, search direction, and step size. During each control interval, (1) a CAE c_i containing the current state x is activated, (2) c_i increments its suboptimal control action $\mathbf{u}_i$ by one step in the search direction, (3) $\mathbf{u}_i$ is applied to the system under control, (4) the resulting IP is evaluated, and (5) c_i modifies its search direction according to this value. (The state is assumed constant during the control interval.) The step size is reduced as the suboptimal control action nears optimum. The accuracy threshold δ_i determines the minimum allowable step size. In some cases δ_i can be shown to be proportional to the radius of r_i. This occurs when the index of performance f is of the form: $f(\mathbf{x},\mathbf{u}) = k\|\mathbf{x}-\mathbf{y}(\mathbf{u})\|$; i.e., f measures the distance between the state "target" vector and an "output" vector y(u) [Gonzalez, 1989].

3 Robotics Application

We describe an application of the on-line learning controller to a dynamic simulation of a 2-degree of freedom robotic manipulator, including gravity and friction effects. The robot's dynamics are unknown to the controller, whose task is to determine the joint torques which optimally move the robot from an initial configuration to a target in a rectangular region in the plane of the robot (Figure 2). The index of performance is a weighted sum of the hand's position and velocity error and the energy and power requirements. It is generally difficult to analytically minimize such a function in the face of the robot's nonlinear dynamics, and in any case the computations necessary will generally preclude real-time control. Likewise, conventional adaptive methods do not produce *optimal* control of such systems.

The control interval is a single *trial*, beginning with a fixed robot configuration and presentation of a randomly-selected target. Since the robot's configuration at the beginning of each control interval is constant, the state $\mathbf{x}=[t_1\ t_2]^T$ consists solely of the 2-dimensional coordinates of the target. The controller computes a 4-dimensional control vector $\mathbf{u}=[\Gamma_{1a}\ \Gamma_{2a}\ \Gamma_{1d}\ \Gamma_{2d}]^T$, consisting of the acceleration and deceleration torques for each of the two joints. (The control vector could as easily have defined the electrical input signal to the joint actuators.) The acceleration torques are applied to their respective joints for the first half of the control interval (0.5 sec), and the deceleration torques are applied during the second half of the interval. The index of performance is computed at the end of the control interval. This value is used by the active CAE to adjust the direction of search.

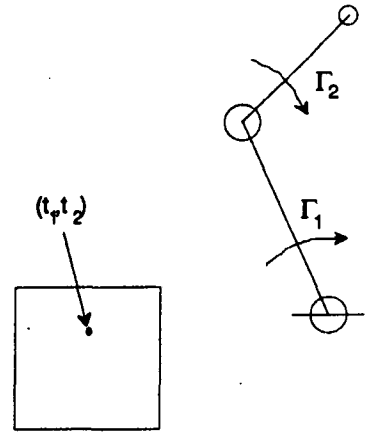

Figure 2. Initial robot configuration and sample target.

Since the control interval coincides with the length of a trial, control during each trial is open-loop, or "blind". During on-line training, the controller constructs a lookup-table containing suboptimal open-loop torque profiles for reaching any target.

The system model consists initially of a single CAE, whose disk-shaped region is centered in the state space. Its control vector has been optimized for a target at the center of state space, and the region radius is correspondingly very small. (The optimization is accomplished by training the initial CAE with a series of trials whose targets are fixed at this location.)

3.1 Simulation Results

Figure 3 shows the CAE regions existing after 10, 100, 1000, and 10,000 trials. When a CAE c_i has a relatively large region r_i, the likelyhood of a (randomly-selected) state x falling in r_i is correspondingly large. Therefore, c_i is activated frequently during on-line operation, and converges quickly to a control vector $\mathbf{u}_i$ satisfying its accuracy threshold. Thereupon r_i contracts. As training progresses the average size of the individual CAEs causes them to be activated rarely, and learning slows.

The size of a region is inversely related to the accuracy of the control action of its associated CAE. Since states are selected uniformly over state space, the variation in size of regions after 10,000 trials is related primarily to the difficulty of optimization in some areas of state space. There also is bias toward greater accuracy at the center of state space due to the influence of the initial CAE, and a bias toward lower accuracy at the edges of state space where some regions overlap the boundary.

After 10,000 trials, the controller succeeds in placing the end-effector on any target with little error. Since energy and power usage are also included in the index of performance, the robot's motion is smooth and efficient. At this point, learning may cease. Alternately, the CAEs may remain active during on-line use of the robot to enable continually-improving performance.

After 10,000 trials, the computer memory requirement for this application is about 893 kb (each CAE requires roughly 450 bytes). This includes the lookup-table of suboptimal control actions (the system model) as well as data for the individual CAE search processes. If learning continues during on-line use, the number of CAEs will increase and memory requirements will continue to grow.

3.1.1 Comparison with Fixed Partition

To compare the learning rate using CAEs versus that using a fixed partition of state space, another experiment is conducted. Each adaptive element is initialized with the control action which is optimum at the center of state space;

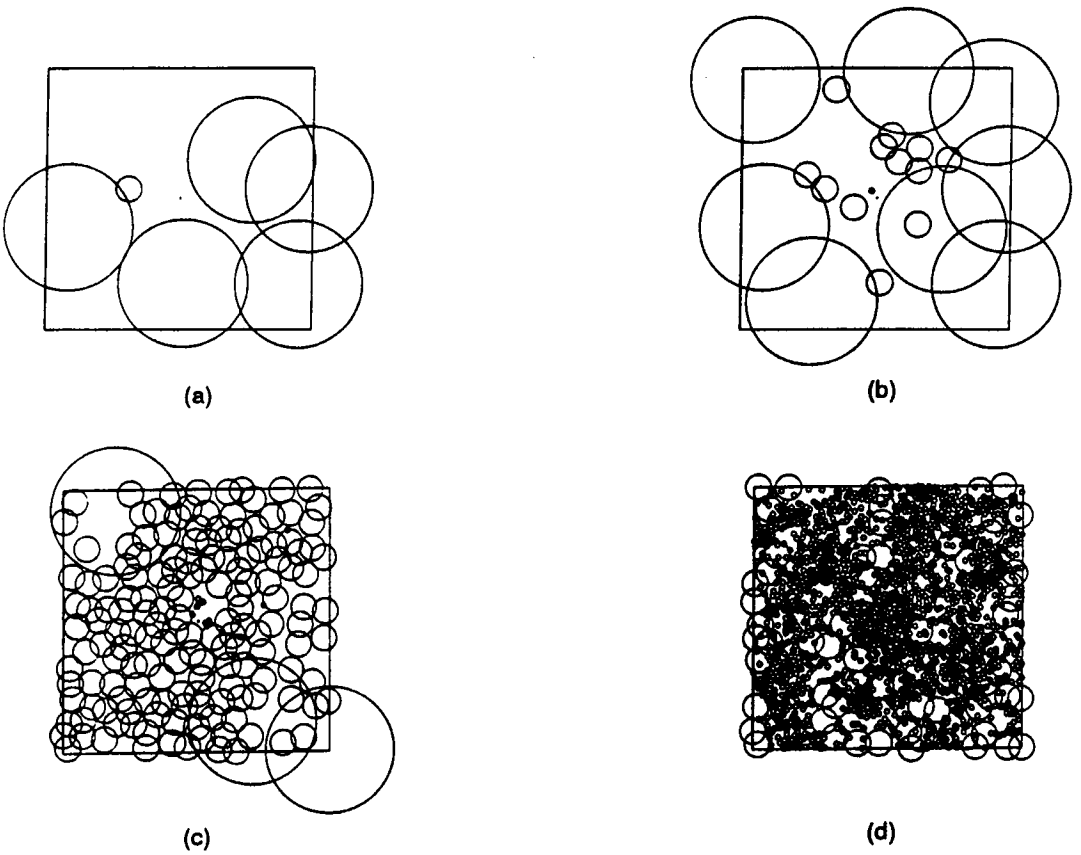

Figure 3. CAE regions after (a) 10 trials, (b) 100 trials, (c) 1000 trials, (d) 10,000 trials.

Trial number	Contractible regions			Fixed region size		
	Number of elements	Average IP	Average region size	Number of elements	Average IP	Region size
10	6	580.7	95.08	10	694.6	5
100	21	441.2	42.52	100	687.4	5
1000	174	243.0	22.70	881	795.2	5
10000	2002	71.3	5.25	4025	611.3	5
100000	10852	41.4	1.78	6390	168.3	5

Table 1. Partition statistics comparing CAE approach versus fixed regions.

i.e., the same value which initialized the original experiment. The radius of each corresponding state space region is fixed to the average radius existing after the original 10,000 trials (5 units). The same 10,000 trials are presented again. As before, adaptive elements are generated whenever a state falls outside of all existing regions. However, once state space is fully blanketed with regions, the partition remains fixed. (The same performance is achieved if this partition is fixed before training begins.)

Table 1 compares the results with those obtained using the CAE approach. Following 10,000 trials, the number of fixed regions is 4025, indicating that the average adaptive element has only been in operation during 2.5 control intervals. Over the course of 100,000 trials the average index of performance of the adaptive elements begins to improve. In contrast, the CAE approach permits learning to begin immediately, and obtains superior performance after only 10,000 trials.

4 Conclusion

The benefits and limitations of this approach for low-level control are apparent from the simulation results. A lengthy training period and large computer memory is required to construct a useful system model, particularly where the state space dimensionality is high. It may be necessary to train the controller off-line using a simulation of the system.

Lookup tables using fixed partitions suffer from lengthy training periods as well. The simulation results verify that the CAE approach achieves greater accuracy with a given training period. The CAE approach is flexible, since the resolution of the partition need not be fixed beforehand and may vary throughout state space. The system model is never "complete", but improves continually during on-line operation.

Efficiency may be improved by utilizing a higher-order (non-constant) approximation to the optimum within the region associated with each search process. Another variation uses a linearized model of the system to help steer search in the appropriate direction.

In cases where the system parameters are changing gradually with time, time may itself be included in the state vector. The effect is to make the distance from the current state to "old" CAE regions greater than the distance to newly-created CAEs, so that the controller can better track time-varying dynamics. To conserve memory, very old CAEs are purged, or "forgotten".

Further improvements in efficiency can be obtained where the overall control problem can be decomposed hierarchically, such that the state spaces relevant to each subproblem have smaller dimensionality than the main state space. Optimization in each of the respective state spaces must be coordinated carefully [Gonzalez, 1989]. This approach exploits the sharing of subproblems among problems, and encourages automatic composition of models in order to model new systems.

While high-level tasks generally require models which more closely represent the environment, these models themselves may be fine-tuned empirically through a meta-learning process based on lookup-table models.

References

[Albus, 1975] J. S. Albus, "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)", *Trans. ASME*, pp. 220-227, 1975

[Gonzalez, 1989] R. E. Gonzalez, *Learning by Progressive Subdivision of State Space*, doctoral dissertation, Univ. of Pennsylvania, 1989

[Rosen *et al.*, 1992] B. Rosen; J. M. Goodwin; J. J. Vidal, "Process Control with Adaptive Range Coding", *Biol. Cybern.*, V. 66, no. 4, Mar. 1992

[Saridis, 1979] G. N. Saridis, "Toward the Realization of Intelligent Controls", *Proc. of the IEEE*, V. 67, no. 8, p. 1122, August 1979

[Waltz and Fu, 1965] M. D. Waltz and K. S. Fu, "A Heuristic Approach to Reinforcement Learning Control Systems", *IEEE Trans. Automat. Contr.*, V. AC-10, pp. 390-398, 1965