# Knowledge of Knowledge and Intelligent Experimentation for Learning Control

**Andrew W. Moore**
The Artificial Intelligence Laboratory, Room NE43-759
Massachusetts Institute of Technology
545 Technology Square, Cambridge, MA 02139
Email: awm@ai.mit.edu

## Abstract

A learning control system observes a plant and uses a computational mechanism to provide future control decisions that improve future performance. An integral component of almost all such schemes is a function learner or classifier [7, 3, 5, 4] which models the world. The debate as to how the function should be learned often overshadows another important feature of a learning control system: how experimentation should take place. The standard method is to use some period of random flailing [7] in order to collect a diversity of experience. In this paper it is shown that if a learning system is able to provide some estimate of the reliability of the generalizations it produces, then the rate of learning can be considerably increased. The increase is achieved by á decision theoretic estimate of the value of trying alternative experimental actions. A further consequence of this kind of learning is that experience becomes concentrated in regions of the control space which are relevant to the task at hand. Such a restriction of experience is essential for continuous multivariate control tasks because the entire state space of such tasks could not possibly be learned in a practical amount of time.

## 1. Learning model-based control

A plant's dynamics can typically be written as

$$\dot{\mathbf{x}} = g(\mathbf{x}, \mathbf{u}) \tag{1}$$

where $\mathbf{x}$ is the *state* of the system, $\dot{\mathbf{x}}$ is the state change or *behavior* and $\mathbf{u}$ is the control input, or *action*. There are thus three components to a dynamic control problem, each a vector. Machine learning can be used to learn the *inverse world model*

$$\mathbf{u} = g^{-1}(\mathbf{x}, \dot{\mathbf{x}}) \tag{2}$$

Thus, at all times, the controller supplies the current state and desired behavior to the learning element, which provides an action which is is hoped will achieve the behavior. The prediction is based on earlier experiences. This method has the advantage of *reactivity*—little computation is required to make a control decision. However, with no experimentation it is in danger of becoming stuck due to initial large errors in the prediction. This is illustrated in Figure 1, in which a learner has made a reasonable fit of a small amount of initial data about the inverse model. The desired behavior of 7.2 is fed into the learner which produces an action of 5. But let us assume that the actual relation between actions and behaviors is the dotted line in the same figure. Then applying action 5 produces behavior 2. This might normally be acceptable for a learning controller, providing future performance improves based on previous mistakes. In this case, however, future performance does not improve because the new piece of data is something we already knew, and the generalization does not change. With a benevolent inverse function (e.g. that of a two-joint robot arm [3]) this might not happen, but in general naive use of a partially learned inverse model is not guaranteed to improve; instead for any significant multidimensional task, many pockets of large inaccuracy such as that above can remain, and never be improved by repeated performance.

The solution to the danger of sticking is to occasionally perform random experiments. Provided each action may be chosen with finite probability, and provided the inverse map is learnable by the learning element being used, then it is guaranteed that the inverse map will eventually be obtained to sufficient accuracy for any task. However, the time taken until successful performance is obtained depends very critically upon the strategy for experimentation. The next section explores various strategies for the mountain car task.

## 2. Random Experimentation Strategies

This section is illustrated by means of a simple dynamic system in which each component is one dimensional. The example, depicted in Figure 2, concerns a car being driven over a mountain range. The horizontal speed $\dot{s}$ depends on the horizontal position $s$ (which determines the current gradient and altitude) and the amount

0-7803-0164-1/91/0000-0683$01.00©1991 IEEE

$u$ by which the pedal is pressed. The dynamics and system limits are given in Table 1. The variables are all scaled into the range $[0.0 \ldots 10.0]$. The control task is to start the car at $s = 0.5$ units and drive it at constant horizontal speed $\dot{s} = 5.7 \pm 0.2$ units. In Figure 3 the behavior (speed) is graphed against state (position) and action (pedal height).

| | | |
|---|---|---|
| $0\text{m} \leq s \leq 1000\text{m}$ , | $-0.15\text{ms}^{-1} \leq \dot{s} \leq 2.5\text{ms}^{-1}$ , | $0\text{cm} \leq u \leq 10\text{cm}$ |
| Height of Road: | $h(s) = 1000P(s/1000)$ where $P(x) = -8x(x^3 - 1.82x^2 + 0.95x - 0.16)$ | |
| Surface Speed: | $v_{\text{surface}}(s,u) = \frac{7}{3}\sqrt{1 - \frac{u}{10}} - \frac{1}{6}\frac{dh}{ds} + h/1200$ | |
| Horizontal Speed, $\dot{s}$: | $\dot{s} = v_{\text{surface}}(s,u)\left(1 + (\frac{\text{dh}}{\text{ds}})^2\right)^{-\frac{1}{2}}$ | |

Table 1: Dynamic equations of the mountain car.

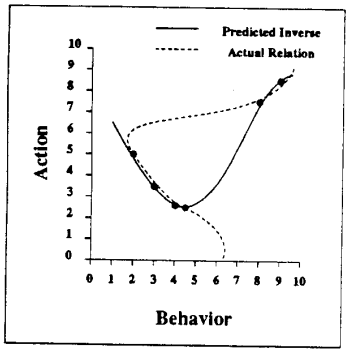


Figure 1: Inverse model generalized from six datapoints. If the learned inverse model is used to achieve behavior "7.2" then action "5" is recommended. When applied, wrong behavior ("2") occurs, a fact that had been observed previously: nothing is learned.

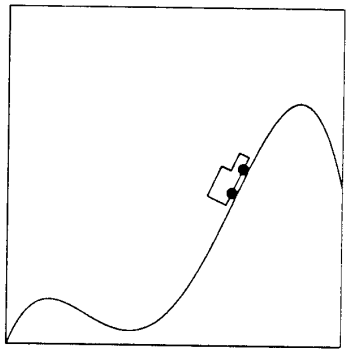

Figure 2: The "Mountain Car": a very simple dynamic system.

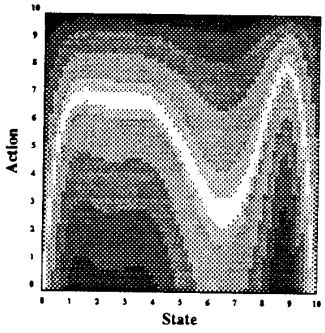


Figure 3: The Perceived State Transition Function of the Mountain car. Brightness denotes the difference in speed from the goal speed of 5.7.

A simple strategy is to have two modes of operation. In *perform* mode, the controller uses the action suggested by the learned inverse model directly. In *experiment* mode, a random action is chosen with no regard whatsoever to the learned inverse model. Figure 4-a graphs the performance (number of incorrect decisions) against trial number of a controller which, on alternate attempts at the mountain car task performs and experiments. Only the data for perform trials are shown. The experiment (which consisted of eighty trials) was repeated forty times. The experimental standard deviations are shown as vertical bars. The inverse model was learned based only on previous experiences, and generalized using nearest neighbor generalization [9, 2, 8], but it is likely that any other generalization method for which fitting to the inverse function was possible would have provided similar results.

In the controller of Figure 4-b, experiments wre not entirely random, but instead consisted of small random modifications to the action suggested by the inverse model. This produced an increase in learning speed, but much time was wasted making the transition from bad action choices to good ones in small steps. A more significant problem with this second mode is the danger that the recommended action is locally best but not globally best.

An improved scheme is "experimentation with checking", shown in Figures 4-c and 4-d. There are no longer two modes of operation. Actions are now always obtained initially by a learned inverse model, but they are not immediately applied. Instead a prediction is made of what will happen if the action is applied. This prediction is by means of a second learning system, which models the forward dynamics. If the forward

prediction is of a behavior within a task-specific tolerance of the desired behavior then the action is used. Otherwise an experiment takes place. Figure 4-c uses random experimentation and Figure 4-d uses local perturbation.

These methods perform better than their predecessors. Local perturbation with checking achieved perfect performance within eighty trials on almost half of the learning runs. This success motivates an attempt to try to squeeze further information from the learning method.

## 3. Probabilistic Experimentation: Using knowledge of knowledge

Let us consider the case where the action suggested by the inverse model has been predicted as inadequate. It is worth stopping to consider what action would best be chosen if unlimited computational resources were available. One possibility is to use the action for which the learned forward model predicts behavior closest to the desired behavior. Such an action could be obtained by searching the forward predictions of all possible actions. However, such a method would be unsuitable because it would favor an action which was reliably predicted to perform mundanely above an action which was unreliably predicted to perform poorly.

A preferable scheme is to find the action which maximizes the ***predicted probability of success.*** This probability is obtained by using not only the prediction of the forward model, but also an estimate of how accurate the prediction is expected to be. For the nearest neighbor generalization, this estimation of accuracy can be derived from the distance of the query to the nearest neighbor. Other learning methods are able to provide measures of the reliability of their predictions. In particular, a neural net which was predicting a forward model could be supplemented with a second net to estimate the densities of previous observations.

The predicted behavior of applying a candidate action in the current state can now be modeled as a random variable. The expected value of the variable is the value predicted by the simple learned forward model. The standard deviation is fixed to depend on the uncertainty estimate. The probability distribution can be fully specified with a standard parametric assumption. Table 2 shows the mathematical analysis in the case where standard deviation is proportional to the distance of the query to the nearest neighbor, and it is assumed that the probability distribution is normal.

Let $B_p$ be a random variable in Dim(**Behavior**)-dimensional space denoting the behavior predicted when we apply the action $\mathbf{a}$ in state $\mathbf{s}$. Let the closest experience (in Euclidian distance) be that when in state $\mathbf{s}^{near}$ the plant used action $\mathbf{a}^{near}$ and produced behavior $\mathbf{b}^{near}$ Then we define $B_p$ to have a normal distribution and:

$$E(B_p) = \mathbf{b}^{near} \qquad Var(B_p) = \sigma^2 = C \mid (\mathbf{s},\mathbf{a}) - (\mathbf{s}^{near},\mathbf{a}^{near}) \mid^2$$

where $C$ is a system constant. Let the goal be to achieve some behavior $\mathbf{b}$ for which

$$\forall i \quad \mathbf{b}_i^{min} < \mathbf{b}_i < \mathbf{b}_i^{max}$$

where $\mathbf{b}_i$ is the $i$th component of vector $\mathbf{b}$. Then the probability of success is

$$\prod_{i=1}^{\text{Dim(Behavior)}} \left( \text{erf}\left( \frac{\mathbf{b}_i^{max} - \mathbf{b}_i^{near}}{\sigma} \right) - \text{erf}\left( \frac{\mathbf{b}_i^{min} - \mathbf{b}_i^{near}}{\sigma} \right) \right)$$

Table 2: Estimating the probability of success of a new action in a new state.

The details of the particular method of estimating the probability of success are less important than the observation that methods such as this which include uncertainty in their predictions provide a well founded statistical analysis of candidate experimental actions. The important properties of such methods are summarized below. Greater detail is provided in [8].

- The qualitative ranking of alternative actions is: (Best) reliably predicted success, (Second) unreliably predicted success, (Third) unreliably predicted poor performance, (Last) reliably predicted poor performance.
- The method only provides a way of ranking alternative actions; not of automatically obtaining the action with the highest probability of success. Instead a small number of candidate actions are randomly

generated, and that with the highest probability of success is selected for use. Empirical results (see later) show that the number of candidate actions generated need not be large for a very large increase in performance quality over that of random experimentation.

- When an action is predicted as nearly successful, it may be worth considering local candidate actions. This is implemented by the following candidate generation algorithm: flip a coin, and half the time choose a random uniformly distributed action and half the time choose an action from a random distribution which is local to the action suggested by the inverse model.

- Let the proportion of behaviors which would be considered successful be $1/\tau$. Analysis under simplified idealized conditions show that under repeated attempts to achieve the same behavior, the scheme in Table 2 is expected to find a successful action in time proportional to $\log \tau$. This compares to a strategy of random experimentation in which the expected time is proportional to $\tau \log \tau$. Empirical results support this analysis.

- It can also be shown that under hostile conditions of many local maxima in the space of actions, this method will not become stuck, although in bad conditions the time to success might become as long as that of random experimentation.

The mountain car task was controlled using the probability of success estimate to choose the most promising of five randomly generated candidate actions. The results are shown in Figure 4-e. The learning rate was significantly faster, and in all forty learning runs perfection had occurred by trial 60.

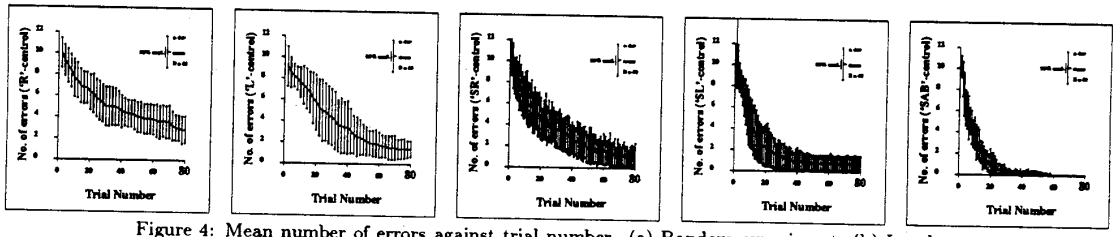


Figure 4: Mean number of errors against trial number. (a) Random experiments (b) Local perturbations (c) Random experiments with forward checking (d) Local perturbations with forward checking (e) Using the probability of success estimate. Error bars denotes standard deviations of forty independent learning runs.

It is interesting to compare the state-action combinations which were experienced using random experimentation with those from this probabilistic experimentation. The state-actions are plotted as dots in two dimensional space in Figures 5 and 6, which show all the experiences during a single learning run. It can be seen that random experimentation fills up the space fairly uniformly, whereas the experiences using the probabilistic method are clustered much more closely to the solution path. This provides an indication of why the probabilistic method is likely to be even more important for higher dimensional problems. Learning should only occur in relevant subspaces of the control space, because it would be impossible, in practical time, to have experiences covering all areas of a high dimensional space.

## 4. Juggling Example

A variety of tests have been carried out with simulated learning control systems in [8]. This section describes a juggling example, depicted in Figure 7. A visually observed ball is bounced on a one-dimensional controllable surface. The ball is observed once each bounce, when it reaches the peak of its trajectory. It then detects the $x$ coordinate of the ball ($x_{\text{top}}$), the $y$ coordinate of the ball ($y_{\text{top}}$) and the horizontal speed of the ball ($u_{\text{top}}$). The vertical speed is zero at the bounce peak. The three values ($x_{\text{top}}, y_{\text{top}}, u_{\text{top}}$) constitute the state of the system.

The surface is always reset to the same height before the ball reaches the top of the bounce. Then, a short period of time, $t_{\text{go}}$, after the ball reaches the top of its trajectory, the surface center starts to move

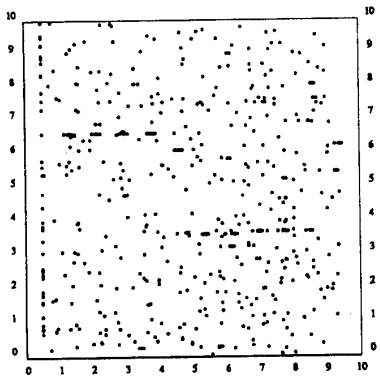

Figure 5: Distribution of experiences in State-Action space after learning with random experiments.

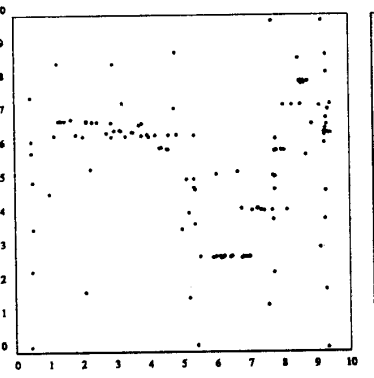

Figure 6: Distribution of experiences in State-Action space after learning with probabilistic experiments.

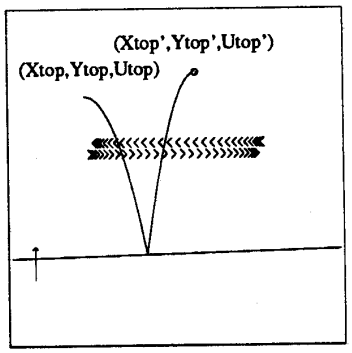


Figure 7: A single ball bounce of the juggling task. The surface was moving upwards, and was at a slight angle.

upwards with constant speed $v_{go}$. The bat is also placed at angle $\theta_{go}$ to the horizontal. Thus the three variables $t_{go}$, $v_{go}$ and $\theta_{go}$ affect the behavior of the surface and constitute the action used to control the ball. Behavior is the ball's next state. 1% simulated random noise is introduced to all variables.

The state, action and behavior spaces are all three dimensional, meaning a six dimensional real valued space of possible state-action experiences. The task was to keep the ball bouncing sixty times without falling outside the visible region, and to follow a predefined series of bounce peaks, shown by the arrows in Figure 7.

If the ball position error before a bounce was greater than a small threshold (5% of the $x$ and $y$ ranges) then the probabilistic method tried to find the action with the highest probability of reducing the error by a factor of two. If the position error was less than the threshold, the search was for the action most likely to keep the error within the threshold.

The task was first attempted by a controller which used the strategy of random experimentation with checking. When the action suggested by the inverse model was predicted to fail a random experimental action was used, drawn uniformly 50% of the time and locally the other 50% of the time. The results were poor: during 200 attempts at the task, in only three cases did the controller succeed in avoiding crashing the ball.

The results for the same experiment using probabilistic experimentation were more successful. These results are shown, and compared with the random experimenter, in Figure 8. The speed and ultimate accuracy can be seen to depend on how many candidate actions are considered. After forty trials the results using 50 candidate actions approached the best possible RMS error, given the 5% position error allowed by the controller.

## 5. Discussion

It has been shown that it is important for a learning controller to use some sophistication in planning useful experiments. It is conjectured that in many cases the learning rate may not be affected so strongly by how accurate the learning method is but by how effective is its experimentation.

Some recent work has attacked the quest for useful experiments in discrete domains from a statistical framework [11, 6, 10]. These approaches attack the harder, and more general, problem of maximizing the long term rewards from an unknown reinforcement signal and require considerably greater computation as well as the assumption of discrete state and action spaces.

If a learning method can provide an estimate of the noise or non-determinism of the forward model, it can also be incorporated into the experimentation strategy. The search for candidate actions can also be aided considerably by a learning method which provides a local gradient estimate of the forward model. When an action is predicted to be close to success, an assumption of local linearity in the forward model can be used to provide a more promising action. In [1] this method is used to great effect in a real ball bouncing task.

Probabilistic experimentation is more computationally expensive during runtime than a simple strategy because it requires several predictions of the effects of several alternative candidate actions. This computa-

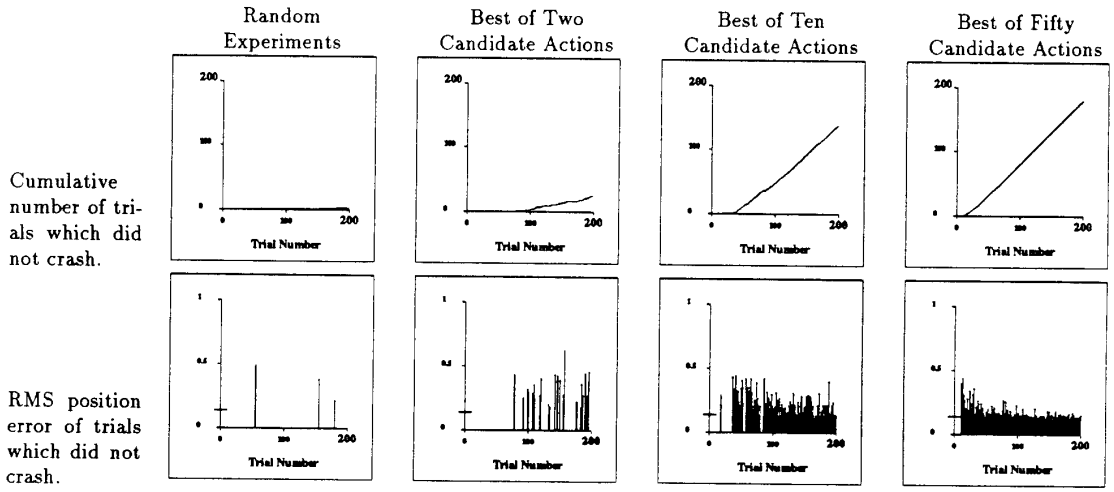


Figure 8: Juggling learning runs. Cumulative successes and accuracies for different experimentation strategies. Each learning run consisted of 200 trials.

tional burden decreases as proficiency is obtained. With increasing frequency, the inverse model provides an action predicted to be adequate and experiments are required with decreasing frequency. This provides an analogy with biological learning of motor tasks which, with increasing practice, require decreasing attention.

# References


[1] E. W. Aboaf, S. M. Drucker, and C. G. Atkeson. Task-Level Robot Learning: Juggling a Tennis Ball More Accurately. In *IEEE International Conference on Robotics and Automation*, 1989.

[2] D. W. Aha. A Study of Instance-Based Algorithms for Supervised Learning Tasks: Mathematical, Empirical and Psychological Evaluations. PhD. Thesis; Technical Report No. 90-42, University of California, Irvine, November 1990.

[3] C. G. Atkeson. Learning Arm Kinematics and Dynamics. *Annual Review of Neuroscience*, 12:157–183, 1989.

[4] A. D. Christiansen, M. T. Mason, and T. M. Mitchell. Learning Reliable Manipulation Strategies without Initial Physical Models. In *IEEE Conference on Robotics and Automation*, pages 1224–1230, 1990.

[5] W. F. Clocksin and A. W. Moore. Some Experiments in Adaptive State Space Robotics. In *Proceedings of the 7th AISB Conference, Brighton*. Morgan Kaufman, April 1989.

[6] L. P. Kaelbling. Learning in Embedded Systems. PhD. Thesis. Technical Report No. TR-90-04, Stanford University, Department of Computer Science, June 1990.

[7] B. W. Mel. MURPHY: A Connectionist Approach to Vision-Based Robot Motion Planning. Technical Report CCSR-89-17A, University of Illinois at Urbana-Champaign, June 1989.

[8] A. W. Moore. Efficient Memory-based Learning for Robot Control. PhD. Thesis; Technical Report No. 209, Computer Laboratory, University of Cambridge, October 1990.

[9] S. M. Omohundro. Efficient Algorithms with Neural Network Behaviour. Technical Report UIUDCS-R-87-1331, University of Illinois at Urbana-Champaign, April 1987.

[10] R. S. Sutton. Integrated Architecture for Learning, Planning, and Reacting Based on Approximating Dynamic Programming. In *Proceedings of the 7th International Conference on Machine Learning*, June 1990.

[11] C. J. C. H. Watkins. Learning from Delayed Rewards. PhD. Thesis, King's College, University of Cambridge, May 1989.