Tracking the evolution of learning on a visual-motor task

Devika Subramanian DEVIKA@RICE.EDU
Department of Computer Science, Rice University, Houston TX 77005

Sameer Siruguri SIRUGURI@RICE.EDU
Department of Computer Science, Rice University, Houston TX 77005

Abstract

How do humans acquire skills on complex visualmotor tasks? From a large sequential corpus of low-level visualmotor data gathered from human subjects, we track the evolution of their control policies in the novice to expert transition. The visualmotor performance data is non-stationary; it is characterized by periods of slow evolution of a policy punctuated by conceptual shifts in which policies change radically. The data is high-dimensional and has non-Gaussian noise. We develop algorithms that build and track models of the policies across the conceptual shifts. The models are "local" and are fitted by a novel extension to locally weighted regression(Moore et al., 1997a). These local models are rich enough to capture individual differences in the task, and are simple enough to learn in real-time. We experimentally demonstrate the effectiveness of our methods by showing the closeness of fit of model performance to human learning curves. We also compare the performance of our methods to decision trees and demonstrate the superiority of local models for the task of learning compact representations of high-dimensional, noisy, non-stationary sequential data.

1. Introduction

Our goal is to build computational models of humans learning to perform complex visualmotor tasks. By a model of human learning, we mean an explicit representation of the human's control policies (mapping from perceptual inputs and internal state to motor actions), and its evolution over time. The models will be used in designing personalized training protocols to help humans achieve high levels of competence on these tasks. This intended use places constraints on the class of models we can consider and the methods for evaluating them. In particular, the models need to be detailed enough to pinpoint opportunities for improving a subject's learning; yet be coarse enough to be unambiguously built from the available visualmotor learning data. Our criterion for evaluating models is empirical: the learning curves of the model and human must match.

In this paper, we design algorithms that build and track models of control policies directly from the low-level visualmotor data. Since verbalization during task performance impairs a subject's ability to acquire the task, the visualmotor data is the only objective source of information from which we can extract the strategy or policy used by the subject. We view the policy extraction task as a compression problem — if there are high-level regularities in the visualmotor data, they will be reliably extracted as policies by our learning algorithms. This approach has the advantage that cognitive modeling constructs arise endogenously from the data, rather than being stipulated *a priori*, as is the practice in most contemporary cognitive modeling efforts.

Our task domain is the NRL Navigation task (Gordon & Subramanian, 1996) developed by Alan Schultz at the Naval Research Laboratory (NRL). It requires piloting an underwater vehicle through a field of mines guided by a small suite of sonar, range, bearing and fuel sensors. Sensor information is presented via an instrument panel that is updated in real-time. The sensors are noisy. Decisions about motion of the vehicle (speed and turn) are communicated via a joystick interface. The task objective is to rendezvous with a stationary target before exhausting fuel and without hitting the mines. A trial or episode begins with the vehicle being randomly placed on one side of a mine field and ends with one of three possible outcomes: the vehicle reaches the target, hits a mine, or exhausts its fuel. Reinforcement, in the form of a scalar reward dependent on the outcome, is received at the end of each episode. Since the mine configurations vary from episode to episode, it is fruitless for subjects to memorize a sequence of actions that will get the vehicle to the target. To solve the task, subjects must learn a policy for choosing actions based on the sensor values presented to them.

The Navigation task belongs to the family of partially observable Markov decision processes. With the addition of the last action taken, we can transform it into a fully observable Markov decision process (MDP). This transformation lends theoretical tractability because deterministic optimal decision procedures exist for MDPs. There are three major sources of complexity in the Navigation task (1) the large number (10^{18}) of distinct sensor configurations for which an action choice has to be computed (2) the large number of potential actions to choose from (153) at each time step, (3) limited binary feedback at the end of each episode, and, (4) a tightly coupled action space in which the different components (turn and speed) cannot be learned independently. Together, these make the task challenging for both our human subjects as well as for reinforcement learning algorithms.

Our learning performance data was gathered as follows. Five subjects ran the Navigation task with a configuration of 60 mines, drift, and low sensor noise.[1] Subjects trained for five days, spending an hour each day running consecutive episodes. The number of episodes per hour varied from around 60 to 160. Each episode varied from 40 to 200 time steps. We collected the time series of sensor action pairs over the five day period. This is the performance data from which we will learn and track the evolution of control policies.

The visualmotor performance data for NRL Navigation is a two-level time series. At the top level, we have a sequence $\mathbf{E}$ of episodes $e_1, e_2, \ldots e_N$, where each episode itself is a time series of form $\{(p_i, a_i) | i > 0\}$. The perceptual input vector $p_i \in \mathbf{P}$ ranges over the discrete set $\mathbf{P}$ (10^{18} in size) of all possible sensor inputs and has 11 components: range,bearing,last turn, last speed and seven sonar readings (s1-s7). The motor output $a_i \in \mathbf{A}$ is drawn from 153 turn and speed choices in the set $\mathbf{A}$. We will extract stochastic policies of the form $\pi : \mathbf{P} \rightarrow \mathcal{P}(\mathbf{A})$. The choice of stochastic policies is dictated by the fact that there is usually more than one "correct" motor output for a given perceptual input vector. We model this aspect by associating a probability distribution over $\mathbf{A}$ for each element in $\mathbf{P}$. Extracting a stochastic policy from the episodic time series above is difficult for several reasons.

1. the high dimensionality of the discrete sets $\mathbf{P}$ and $\mathbf{A}$. In particular, if we treat the problem as one of classification, we have 153 different output classes to learn. We provide a visual demonstration in Figure 1of the variation in action choice distributions over different subsets of $\mathbf{P}$. That is, the output class distribution varies a lot between regions of the input space. A learning method has to implicitly or explicitly capture the right equivalence classes in $\mathbf{P}$ to be able to accurately generate actions from these widely differing output distributions.

2. non-stationarity of the policy, since it changes with training. This property is illustrated in Figure 2 where we show the distribution of action choices for Subject 1 on days 1 and 4 of training for the class of situations where the subject is close to mines.

   Let $\pi_{i,j} : \mathbf{P} \rightarrow \mathcal{P}(\mathbf{A})$ denote the policy extracted from contiguous episodes $e_i, \ldots, e_j, N \geq j > i$ of $\mathbf{E}$. We need to partition $\mathbf{E}$ into n maximal, non-overlapping contiguous segments $[1, i_1], [i_1 + 1, i_2], \ldots, [i_{n-1}, N]$ which span the interval $[1 \ldots N]$ such that the policy is stationary over each segment.

3. non-white noise in the data caused by joystick hysteresis and lapse of a subject attention. Such noise is particularly hard to deal with in the context of non-stationary data, since we need to distinguish the case when there is more than one correct motor output for a given perceptual input p, and when one or more of the motor outputs associated with p in the data are incorrect.

In this paper, we develop an algorithm that partitions $\mathbf{E}$ into nearly stationary segments using derivatives to track when the change in policy between adjacent segments is significant. We then build "local" models of the stochastic policy over the stationary segments. The models are learned by a novel extension to locally weighted regression(Moore et al., 1997a) called *biased dimension elimination.* We experimentally demonstrate the power of biased dimension elimination in coping with the high dimensionality of the set $\mathbf{P}$. These local models are rich enough to capture individual differences in the task, and are simple enough to learn in real-time. We experimentally demonstrate

[1] Five undergraduates at San Diego State University participated in this experiment. The data collection was performed by our collaborators Diana Gordon and Sandra Marshall.

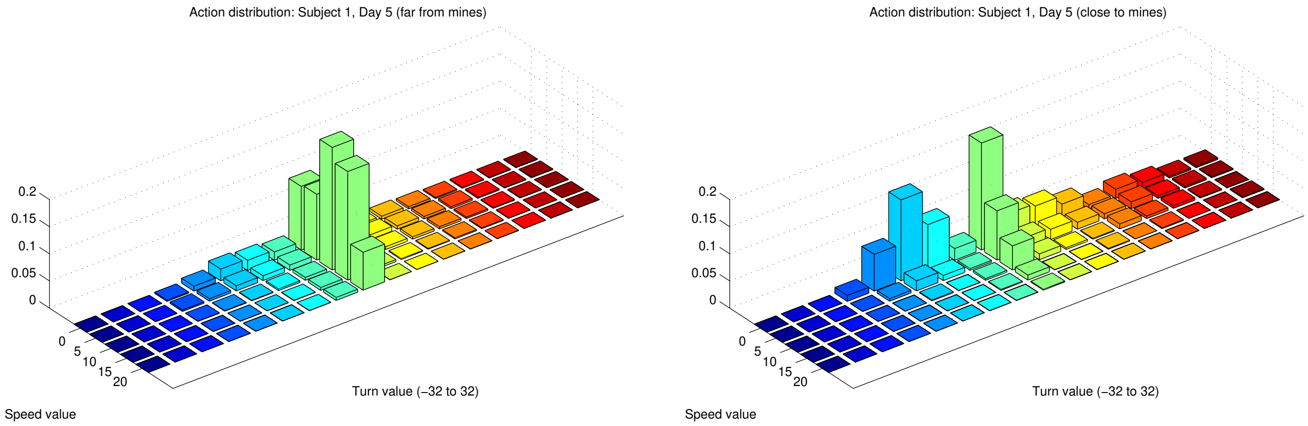

Figure 1. Action distributions, Subject 1/Day 5, for (a) far-from-mine and (b) close-to-mine situations

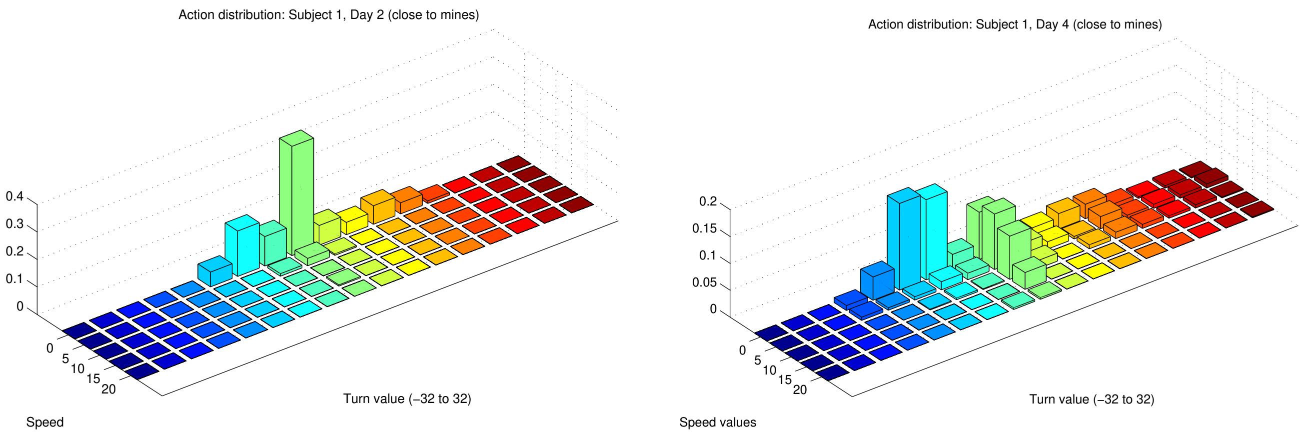

Figure 2. Action distributions, Subject 1, for close mine situations: (a) Day 2 and (b) Day 4

the effectiveness of our methods over other local learning methods (locally weighted regression and weighted k-nearest neighbor) by showing the closeness of fit of our model's performance to human learning curves. We also compare the performance of our methods to a global inductive learning technique (decision trees) and demonstrate the superiority of local models for the task of learning compact representations of high-dimensional, noisy, non-stationary sequential data.

The paper is organized as follows. In Section 2, we present the segmentation algorithm based on policy derivatives. We motivate the parametric structure of our models of the stochastic action choice policies in Section 3 and describe the biased dimension elimination extension to locally weighted regression. In Section 4, we describe our experimental protocol for measuring the effectiveness of our model-building algorithms relative to other local and global learning algorithms. We then present our primary experimental results. We conclude in Section 5 with a discussion of our experimental results and place our work in the context of existing work on learning policies from input-output data.

2. Segmentation by computing policy derivatives

The essential idea is to process episodes in blocks of size w and learn stochastic policies from each of them. Given a distance measure over the space of policies, we can compute the distance between policies over two successive blocks. We study how this distance changes with time (i.e., block number) and use a standard threshold of the mean plus twice the standard deviation on the change in policy distance between successive blocks to identify blocks where there is a significant change in the action policy. The choice of w is dictated by the two competing factors. Making w

as large as possible allows us to construct policies that span $\mathbf{P}$ better, which as we shall see, reduces the complexity of calculating the distance between two policies. Making w as small as possible allows us to pinpoint the location of the policy shift more accurately. We experimentally determine the best value of w for our data to be 20.

More formally, let $\pi_{i,i+w}$ be the stochastic policy derived from episodes $e_i \dots e_{i+w} \in \mathbf{E}$ and let $\pi_{i+w+1,i+2w}$ be the stochastic policy derived from the next block of w episodes, $e_{i+w+1}, \dots, e_{2w}$. Suppose we have a distance function Δ_P over the space of all policies. Then the policy derivative at the block of width w that starts at episode i is

$$\delta(i, w) = \frac{\Delta_P(\pi_{i,i+w}, \pi_{i+w+1,i+2w})}{w}$$

To design an appropriate Δ_P, we first need to fix a representation for the policies. We consider local policy models of the kind created by nearest neighbor methods and locally weighted regression: a lookup table of the form $\{p, \mathcal{P}(\mathbf{A})\}$ where p ranges over the perceptual inputs seen in the given block of w episodes, and $\mathcal{P}(\mathbf{A})$ is the distribution of motor outputs associated with p in the same block of w episodes.

For this representation of policies, a natural distance measure is to compute the average distance between action distributions associated with each p that occurs in the union of the two policy lookup tables. We choose KL-divergence of two discrete distributions as a measure of the distance between them. For lookup tables with 10^3 to 10^4 distinct p's, it is impractical to compute the average distance over all p's. To get around it, we compute the average distance over a small number of randomly chosen p's from the policy tables. We experimentally determine that repeatedly and randomly sampling 5% of the p's from each lookup table keeps the variance of the distance measure obtained under 1% of the mean of the distance distribution. In Figure 3 we use this Monte-Carlo sampling technique to rapidly estimate the policy derivative for Subject 1 on the 4th day of training.

There are only three policy derivative peaks higher than the cutoff of the mean plus twice the standard deviation of the distribution of derivative values for Day 4. Overlaid on the policy derivative measure is the learning curve of the subject. Note that there are three plateaus of performance corresponding to episodes 200 to 240, 260 to 300 and 350 to 400. These plateaus represent the unchanged performance profiles and we associate them with the use of a fixed policy. Our derivative measure accurately picks out the rising edges of

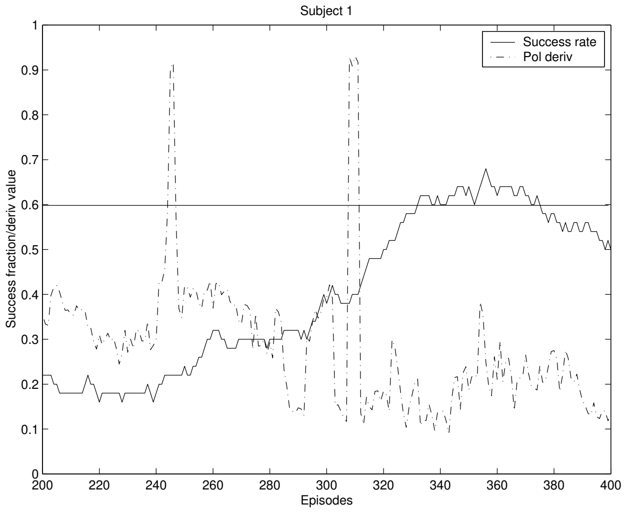

Figure 3. Policy derivative overlaid on success rate (Subject 1)

the performance graph between these two plateaus, where the subject's performance undergoes major improvements, which we believe are associated with significant modifications to the action policy. Consistent with predictions from the cognitive science literature, we find that there are only a few shifts in strategy or policy in human subjects learning this task.

3. Learning models of control policies

Now that the episodic data is segmented into nearly stationary segments, we learn models that map perceptual inputs to distributions over actions using data from each of these segments. The first model we construct is simply a lookup table which contains for each observed perceptual input vector in the episodes, a distribution of the actions that were taken in response to it. Given a perceptual input vector p in the lookup table, we output an action drawn from the stored action distribution associated with it. If p is not in the lookup table, we locate its 100 nearest neighbors using the L_2 norm of the difference between the two perceptual input vectors as the "distance function".[2] To facilitate the search for the 100 nearest neighbors, the perceptual input vectors are stored in a kd-tree. We used ANN, a program that performs nearest-neighbour searches, which we modified to handle "wrap-around" dimensions. The kd-tree representation of the stochas-

[2]The bearing dimension is treated as a "wrap-around" dimension. That is, we pick the minimum between the absolute difference between the elements and the upper limit of the dimension (12) minus this absolute difference.

tic policy lookup table occupies about a megabyte of memory, which is a extremely small compared to the estimated size of $\mathbf{P}$ (10^{18}).

We then investigated two methods for computing an action for input vector p using these nearest neighbours:

1. **Weighted Averaging:** An action is computed as the weighted average of the actions taken for each of the nearest neighbours, inversely weighted by the distance to the the vector p.

2. **Locally-weighted regression:** We fit a surface to the neighboring perceptual vectors using a distance weighted regression (Moore et al., 1997b).

Both methods use the same distance function, viz. the L_2 norm of the difference between two perceptual input vectors, where each dimension is scaled to be in the range $[0, 1]$. The kernel width for locally weighted regression is set to assign a weight of 0.01 to the neighbour that is twice as far away from the query as the nearest neighbour.

4. Testing the learned models

To test the performance of our models against that of the subject, we need a standard train/test protocol. Devising a fair protocol for testing prediction accuracies over of a non-stationary data source is extremely difficult. This is because wholesale revision of policies at the strategy shifts need to be taken into account for properly formulating the "train" and "test" sets. Fortunately, by day 5, most of our subjects have no major shifts in policy, so we use a standard cross-validation protocol to test the fit of the models to the learning curve of the human subjects. We divide the data for day 5 into 10 chunks and build a model out of 9 of the chunks and test it on the left out chunk. The process is repeated 10 times and the average performance of the models (measured as number of successful episodes in a given window) is reported. We ran this train/test protocol on all of our subjects. However, to save space, we only show the results on two of our subjects — Subject 1, who trained to almost 90% success on the task after 5 days, and Subject 2, who achieved only 30% success after the same amount of training.

4.1 The need for richer policy models

It became clear experimentally that the performance of models that calculate action based purely on the current perceptual vector do not match human performance at all (see Figure 4). This is because 0-gram models are not rich enough to capture the action choice basis of our subjects. By analyzing the situations where 0-gram models generated actions inconsistent with our human subjects, we discovered that action choice in situations very close to mines was a function of a prior action history of 4 (the action at time $t-1$, $t-3$, $t-5$ and $t-7$), and the difference between the perceptual vectors at time t and $t-1$. Thus for a very small fraction $\mathbf{M}$ of the perceptual input vector space $\mathbf{P}$, a purely reactive stochastic policy $\pi : \mathbf{P} \rightarrow \mathcal{P}(\mathbf{A})$ is insufficient to generate the right action distributions. Our local models therefore have a two-tier structure. Over all but a small subset $\mathbf{M}$ of $\mathbf{P}$, we learn a reactive stochastic policy that calculates motor output on the basis of the current perceptual input alone. However, for the subset $\mathbf{M}$ (corresponding to situations which forms less than 2% of the size of $\mathbf{P}$), we capture more state to accurately predict the action choice behavior of our subjects. Each of these policy components is conceptually represented as a lookup table, and physically stored as a kd-tree.

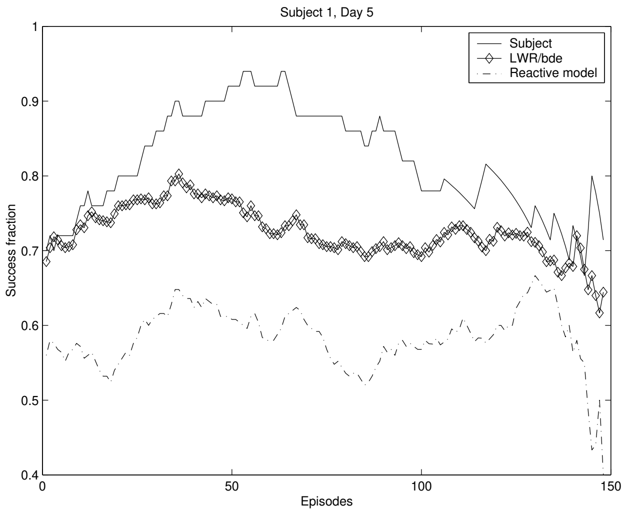

Figure 4. Comparison with 0-gram model (Subject 1/Day 5)

4.2 Comparison of locally weighted regression and Weighted averaging

The performance of the two tiered policy is shown in Figures 5 and 6 for the two subjects identified above. Note that locally weighted regression performs worse than weighted average for both subjects. When we investigate why this is so by replaying the episodes played by the model, we observe that for some $p \in \mathbf{P}$, the regression returns a low turn and a zero speed action pair. Thus the perceptual input vectors remain the same (since the sub is not moving forward), thus forcing a loss by a timeout. The combination of low turn and zero speed values are computed by locally weighted regression for these p vectors, because the nearest neighbours are all on one side p in the (11-)dimensional perceptual input vector space and hence, even though the action responses for all the neighbours have high turn and speed values, the predicted response itself is extrapolated to a pair of low values.

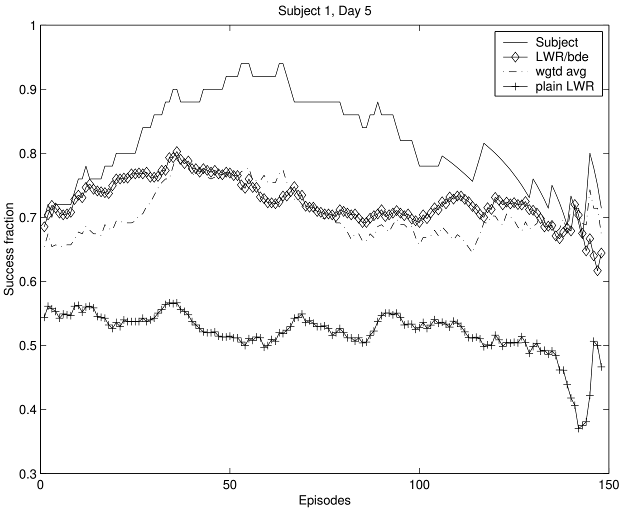

Figure 5. Comparing LWR/BDE with other methods (Subject 1, Day 5)

4.3 Improving locally weighted regression by biased dimension elimination

To fix this problem, we introduce *biased dimension elimination*. The idea is that if we delete all those dimensions from the neighbouring vectors whose values over the neighbours are all greater or all less than the value of that dimension in p, then the regression is guaranteed to interpolate the response instead of extrapolating it. Algorithm 1 describes the procedure for deleting the offending dimensions. We conjecture that the procedure might have a cognitive equivalent in that interpolation is more easily performed mentally than extrapolation and hence, when presented with vectors containing biased dimensions (as defined above), the subject will choose to ignore the biased dimensions rather than spend time calculating the correct extrapolation.

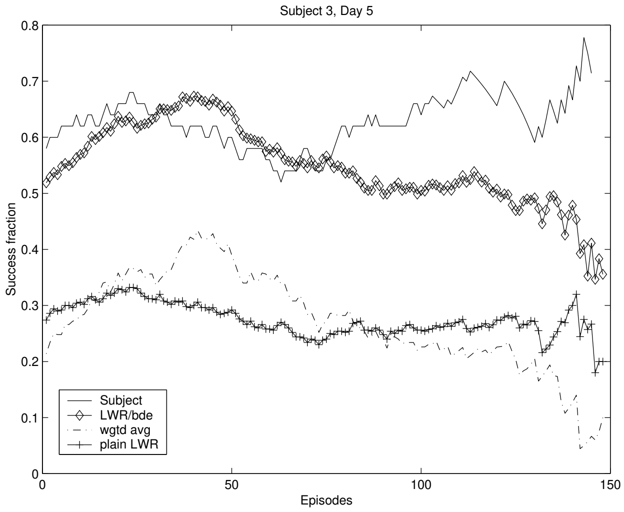

Figure 6. Comparing LWR/BDE with other methods (Subject 3, Day 5)

Algorithm 1 (**Biased dimension elimination**) Given a set of vectors P and a query q, return the set with dimensions biased wrt q eliminated

for each dimension d **do**
  $P' \leftarrow$ set of points in P that receive a weight $>= .01$
  biased_dim $\leftarrow$ NOT($\exists p_1 \in P'(p_1[d] < q[d])$ AND $\exists p_2 \in P'(p_2[d] > q[d])$)
  level_dim $\leftarrow$ ($\forall p \in P'(p[d] = q[d])$)
  if biased_dim AND NOT level_dim **then**
    Delete the d^{th} entry from all $p \in P$.
  end if
end for
if all dimensions were deleted **then**
  Restore all dimensions.
end if

4.4 Evaluating the performance of biased dimension elimination

Figures 5 and 6 summarize how well locally weighted regression with biased dimension elimination on the two tier policy representation match human learning curves. It is surprising to see this degree of fit between a simple model learned from the subject's performance data and the subject himself. Note that we are able to capture individual differences in learning with the same model. Our models are simple to construct and they can be built in real time, making them very useful for shaping training of subjects. A comparison of the action distribution associated with a particular perceptual input vector learned from the subject and that of a machine learned optimal policy can be used to advise subjects about correct actions for those inputs. To further improve the fit to human performance, we need to enrich policy representations and automatically learn the subset **M** of inputs for which more history is needed.

4.5 Comparison against decision tree representations of policies

We learned decision tree representations of the policy function $\pi : \mathbf{P} \rightarrow \mathcal{P}(\mathbf{A})$ from the stationary episode segments as in (Sammut et al., 1992). The performance of this global technique is shown in Figure 7. The decision trees are very large (with over 10,000 nodes) — they are unable to learn appropriate partitions of the perceptual input vector space **P**. This demonstrates the fact that for our data, local models are better fits than global ones. We conjecture that this holds because of the wide variation in output class distributions between the various regions of **P**. Augmented with a good clustering technique that creates output-equivalent partitions on **P**, we expect to have better fits to global models such as the ones learned by decision tree.

5. Conclusions

The ability of the two-tiered lookup table model to achieve a good fit to human learning even with a sparse sampling of the perceptual input space, suggests that it is a good model of human learning. The model is rich enough to capture individual differences in the task, and is yet simple enough to learn in real-time. Our segmentation technique convincingly demonstrates the presence of policy shifts in the evolutions of our subjects' learning skills. The main open problem we have is to devise even more compact models that can be verbally conveyed to our human subjects. We are currently working on techniques that compress our lookup table models by statistical means.

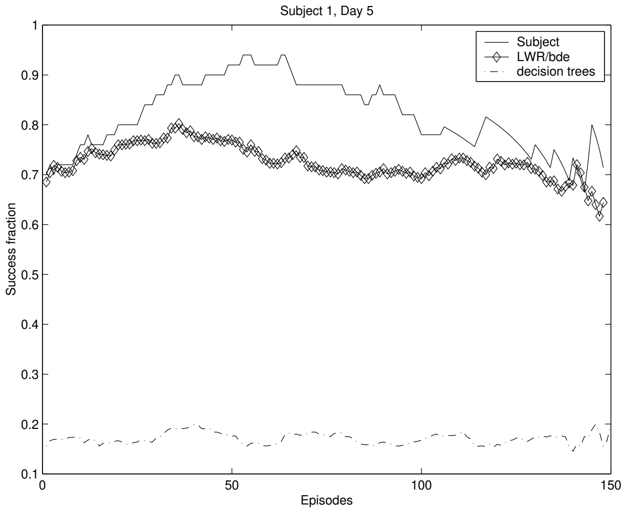

Figure 7. Comparison of LWR/BDE with decision trees (Subject 1, Day 5)

References

Gordon, D., & Subramanian, D. (1996). Comparison of action selection learning methods. *Proceedings of the Third International Workhop on Multistrategy Learning, Harpers Ferry* (pp. 95–102). Fairfax, VA.

Moore, A. W., Atkeson, C. G., & Schaal, S. A. (1997a). Locally weighted learning. *AI Review*, *11*, 11–73.

Moore, A. W., Atkeson, C. G., & Schaal, S. A. (1997b). Locally weighted learning for control. *AI Review*, *11*, 75–113.

Sammut, C., Hurst, S., Kedzier, D., & Michie, D. (1992). Learning to fly. *Ninth International Conference of Machine Learning* (pp. 385–393). Aberdeen: Morgan Kaufmann.