
Simple Learning in Moded Non-stationary Environments

Michael L. Littman

AT&T Labs–Research, 180 Park Ave. Room A275, Florham Park, NJ 07932-0971 USA

MLITTMAN@RESEARCH.ATT.COM

Sushant Agarwal

zzz Chicago, IL

SUSHANT@ALUMNI.DUKE.EDU

Abstract

Non-stationary environments are an important application domain for learning algorithms. This paper examines a simple type of prediction problem in which an environment changes over time according to a Markov process. We motivate the study of this type of environment and show how the simple approach of learning using a fixed learning rate can do well, simultaneously tracking changes between modes and smoothing internal variability within modes. We show that the ideal learning rate varies from environment to environment and present an algorithm that automatically identifies a good learning rate. We evaluate this approach over a set of simulated examples.

1. Introduction

One of the most compelling reasons for including learning algorithms in engineered systems is to allow them to automatically adjust their behavior to changes in their operating environment. Adaptation in such non-stationary environments is critical for robust systems.

The present research is focused on understanding the behavior of a class of very simple learning algorithms in some very simple non-stationary environments. Instead of studying arbitrary kinds of environmental change over time, we restrict our attention to “moded” change: the parameters controlling the environment are stable for periods of time, then change suddenly to some other setting. A formal model of this type of environment called a *hidden mode Markov decision process* (HM-MDP) was proposed by Choi et al. (2000).

In this investigation, we look at the prediction version of HM-MDPs. That is, we try to predict a single real-

valued quantity drawn from an unknown (and perhaps changing) distribution. This problem comes up when trying to apply value prediction algorithms such as Q-learning (Watkins & Dayan, 1992) to HM-MDPs and we believe it is also an interesting problem in its own right.

In spite of their importance, non-stationary environments have not received a great deal of attention in the reinforcement-learning community. There are a few notable exceptions. Dayan and Sejnowski (1996) describe an algorithm that uses a known model of the environmental change to make decisions. The algorithm averages all possible states of the environment and plans in the resulting “certainty equivalent” environment. This helps tame the massive computational complexity that comes from considering all possible future environments independently. When applied to our moded prediction problem, this algorithm reduces to the Bayesian model-based algorithm we study in Section 4.1.

Choi et al. (2000) describes how to learn an HM-MDP model by interacting with an environment. The resulting model can then be used as a restricted type of partially observable Markov decision process and a suitable plan identified. Both learning the model and planning in the model can be quite challenging, which is why we elected to study simple approximation algorithms in the present work.

Stone (2000) shows how a system can learn multiple policies for an HM-MDP by giving the system an extra feature that can be used as a noisy indicator of the hidden mode. Littman and Ackley (1991) show how multiple levels of adaptation can be used to track a slowly changing environment.

2. Example Environments

Moded non-stationary environments are not the most general form of environment change, but they appear to be quite common. To help motivate the study of this class of environments, we present a set of hypothetical examples:

- A car with a transmission that adapts to a driver’s habits may have two or more drivers. Consider the case of a vehicle, used by two people, that tries to predict the next incoming reading from the feedback system embedded in the transmission. The different habits of the two drivers result in the the feedback received by the learning system being separated broadly into two sets, or modes of readings.
- A news website’s day-time traffic patterns can change based on events and breaking news. Note that this is a different type of non-stationarity from time-variant fluctuations, such as the predictable rise in traffic every morning or around lunchtime, which can be predicted if the system clock is included in the state space. Predicting web site response time, then, can be moded in this sense.
- Small scale mobile robots turn differently on hard floors and carpets. If the robot cannot directly sense the floor type and is mainly driven based on local features (move toward the light), then predicting the angle of turn produced by a given force is well modeled as a moded non-stationary environment with “floor” and “carpet” modes.

At a high level, we have identified three basic categories of moded environments: distinct multiple users (different drivers of the same car, different users of the same software system), distinct multiple environments or tasks (using a robot in different places, using a portable computer on the road or at a desk), or unpredictable short-term changes (special events on a network, traffic jam).

As an example of the second category of environment, we collected information on interrupt frequencies on a computer as its user carried out a number of distinct tasks (which we logged) during a typical 24-hour period. Figure 1 gives the distribution of interrupt frequencies for two distinct modes: away from the computer (left) and programming (right). The multiple peaks in the “away” distribution correspond to different screen savers. Because the means of the distributions are different, effective prediction in this tasks requires tracking the mode changes.

3. Problem and Notation

We model a moded environment as consisting of m modes $1 \leq i \leq m$. Each mode i consists of a random variable X_i with mean $\mu_i = E[X_i]$ and variance $\sigma_i^2 = E[(X_i - \mu_i)^2]$.

Abusing notation, we write X as the random variable corresponding the actual sampled values. That is, when the mode is i , $X = X_i$. The identity of the current mode evolves according to the mode-transition function λ_{ij} . In particular, the probability that the mode is j at time $t + 1$ given that the mode at time t is i is λ_{ij} . We make the assumption that $\lambda_{ii} \geq 1/2$; that is, the mode is more likely to stay the same than it is to switch. This is intended to capture the idea that modes are relatively stable and to provide an opportunity for meaningful adaptation to a mode before it switches. In fact, for most applications, we imagine that λ_{ii} is very close to 1.

The objective is to try to predict the sample x of X at each timestep. Prediction error is defined to be the expected squared difference between the predicted value and the sampled value x .

Note that, mathematically, this model is simply a hidden Markov model with real-valued observations and high probability self-transitions.

zzz A closely relate model replaces the real-valued observations with a finite set. Each mode has a multinomial distribution over observations and the objective is to minimize prediction probability.

zzz Sushant: A concrete example might be button pushes on an elevator. In the morning mode, there is some distribution over selected floors, corresponding roughly to the occupancy of the building. In the evening mode, the floors become less likely and the lobby overwhelmingly more likely. Do you think it might be interesting to create a second set of graphs for this “discrete” case?

4. Algorithms

Given that many applications involve moded behavior, we’d like a robust way of predicting value that achieves low error, even in the face of mode changes.

4.1 Bayesian model-based approach

At one extreme, an algorithm could try to learn the underlying model and use it for prediction. We call this the *Bayesian model-based approach*. Given a complete model of the environment, the algorithm needs to track the mode and can use this state estimation to

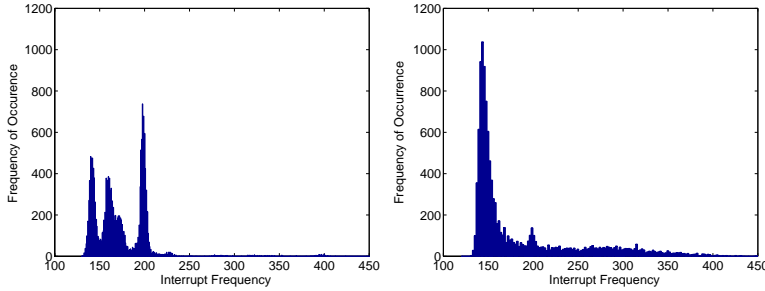


Figure 1. Modes from a 24 hour study of interrupt frequency on a desktop computer

make optimal predictions.

At time t , the algorithm has a probability distribution over modes: ρ_i is the probability of the environment being in mode i at time t . To minimize the error, the algorithm should predict the mean of the resulting distribution: $\sum_i \rho_i \mu_i$.

When the sample x arrives, the algorithm must use this information to derive ρ' , an updated probability distribution over modes. Via Bayes rule: $\rho'_j \propto \sum_i \rho_i p(x|i) \lambda_{ij}$, where $p(x|i)$ is the pdf for mode i and the constant of proportionality is set to make $\sum_j \rho'_j = 1$. For example, for a normal distribution $N(\mu_i, \sigma_i)$, we have density

$$p(x|i) = \frac{1}{\sigma_i \sqrt{2\pi}} e^{-\frac{1}{2} \left(\frac{x - \mu_i}{\sigma_i} \right)^2}.$$

The parameters of the model, such as $p(x|i)$ and λ_{ij} , could be learned from experience using a variant of Baum-Welch (Choi et al., 2000).

4.2 Fixed learning-rate approach

A standard approach to using learning for prediction is to keep an estimate \hat{x} , which is used as the prediction whenever it is needed. When a new value x is sampled, the update

$$\hat{x} \leftarrow (1 - \alpha) \hat{x} + \alpha x \quad (1)$$

is used to create a new estimate.

Here, $0 \leq \alpha \leq 1$ is the learning rate or stepsize parameter. If it is moved to zero at an appropriate rate, the estimate becomes the optimal non-adaptive prediction. In particular, the fixed prediction that minimizes prediction error is the mean of the distribution derived by assuming that the mode is chosen according to the stationary distribution of the Markov chain formed by λ . Let π_i be the stationary probability of the environment being in mode i . Then, \hat{x} converges to $\sum_i \pi_i \mu_i$.

Decaying the learning rate to zero, while sensible in that it results in convergence to the ideal fixed estimate, fails to take advantage of the changes in mode over time. An extremely natural approach is to simply leave the learning rate at some fixed value. This allows the learning algorithm to continue to adapt to mode changes throughout, although it inhibits convergence.

zzz There is almost certainly related work to cite here!

When using a *fixed learning rate*, there is a tension between two principle sources of error: between-mode error and within-mode error. Within-mode error results from predicting a value that is not the mean of the current mode. To reduce this error, the learning rate should be pushed toward zero to drive the prediction to the mean. On the other hand, between-mode error comes from not responding quickly to a mode change and predicting a value influenced by a previous mode. To reduce this error, the learning rate should be pushed toward one to reduce the reaction time to a mode change.

By varying α between zero and one, we have a one-dimensional family of policies that are appropriate for non-stationary environments. There are several important questions that follow from this observation:

- For any given environment, is there always a value of α that works well?
- Does the best setting for α vary from one environment to another?
- Is there a simple way to set α to a good value without knowing a model of the environment?

We argue that the answer to all these questions is “yes.”

5. Simulation Results

Is the optimal value of α fixed or is it a function of the environmental parameters? To explore this issue, we

ran a set of simulations using different parameters and varying α from 0 to 1. These are two-mode environments, with observations selected from Gaussian distributions. Figure 2 presents the learning rate against prediction error obtained by systematically varying the variance within modes ($\sigma_2 = 2$ to 20), the separation between modes ($\mu_2 = 15$ to 40), and the probability of switching between modes ($\lambda_{12} = \lambda_{21} = 0.03$ to 0.5), one at a time. The basic parameters defining the simulations were $\mu_1 = 10$, $\sigma_1 = 3$, $\mu_2 = 25$, $\sigma_2 = 3$, $\lambda_{12} = \lambda_{21} = 0.01$. Points are averages over 6 runs of 2000 steps. All values were generated from normal distributions.

In the plots, the minimum of each curve (optimal learning rate) is marked with a diamond. The top lefthand graph in Figure 2 shows the ideal setting for α decreasing with increasing variance (SD), since this increases the relative magnitude of within-mode error. The top center graph in Figure 2 shows the ideal setting for α increasing with increasing separation between the mode means, since this increases the relative magnitude of between-mode error. The top righthand graph in Figure 2 shows the ideal setting for α first increasing, then decreasing with increasing probability of mode switch. The initial increase reflects the increase in the relative magnitude of between-mode error as mode changes become more common. After a point, however, mode switches become so common that the learning algorithm is unable to track the changes. At this point, the two modes have become merged, and within-mode (merged) error dominates—the algorithm should simply predict the stationary mean.

zzz Sushant: These could probably be bigger. The top right graph is very hard to read, I think. Which line is which? Also, when we say comparison is to optimal, did we actually do Bayesian tracking, or did we use the “cheat” discussed in the analysis section? If the latter, we should say so.

These experiments show that the mapping from environmental parameters to optimal α is complex. The bottom row of graphs gives the same data relative to the performance of the Bayesian model-based algorithm. In no case is the best value of α worse than 1.4 times the optimal.

6. Fixed Grid Method

The previous section suggests that the fixed learning-rate approach can work very well given the correct α —but, how can we know the ideal α to use, given that it changes depending on the parameters of the

environment?

The *fixed grid method* is an approach to solving this problem, and is based on maintaining and updating a fixed number $N + 1$ of predictors \hat{x}^n of the incoming value at each time step, where $0 \leq n \leq N$. Predictor n is calculated using the fixed learning-rate approach-method (Equation 1) with $\alpha = n/N$.

At the same time, it keeps a running estimate of the mean squared error for each of these estimates:

$$\epsilon^n = \frac{\sum_{t=0}^T (\hat{x}_t^n - x_t)^2}{T},$$

where \hat{x}_t^n is the estimate for predictor n on timestep t , x_t is the sample on timestep t , and T is the number of elapsed timesteps. Note that this quantity can be updated efficiently by keeping a running sum of error.

To use this information to produce an estimate, the predictor with the least mean squared error ϵ^n is used. Results for a representative set of 2-mode environments are given in Table 1. Results are based on a single run of 5000 steps.

zzz Sushant, is there any way to find out the best value of alpha for each of the lines of this table? I think this would be a useful column to include in the table.

The fixed-grid approach behaves nearly identically to the fixed learning-rate approach for the best α from a fixed set without any knowledge of the environment in advance. The accuracy with which the optimal learning rate is captured depends on the grid density selected by varying the number N of predictors. There is a tradeoff, therefore, between running time and prediction error.

7. Competitive Analysis

zzz need to rework this.

In this section, we show that, for 2-mode environments, the error obtained using a fixed learning rate of $\alpha = 1$ is no more than twice that of the optimal algorithm. This is a form of *competitive analysis*. As before, we assume self-transitions occur with probability at least one half.

In a truly unknown environment, an optimal strategy would be simultaneously learning about the environment and using this information to make predictions, making analysis difficult. However, we can get a lower bound on the error of the optimal strategy by analyzing an algorithm that uses “insider” information. Imagine that the algorithm knows the model (distribution of the X_i s and the mode transition probabilities λ_{ij}). Further, each time the algorithm is given the

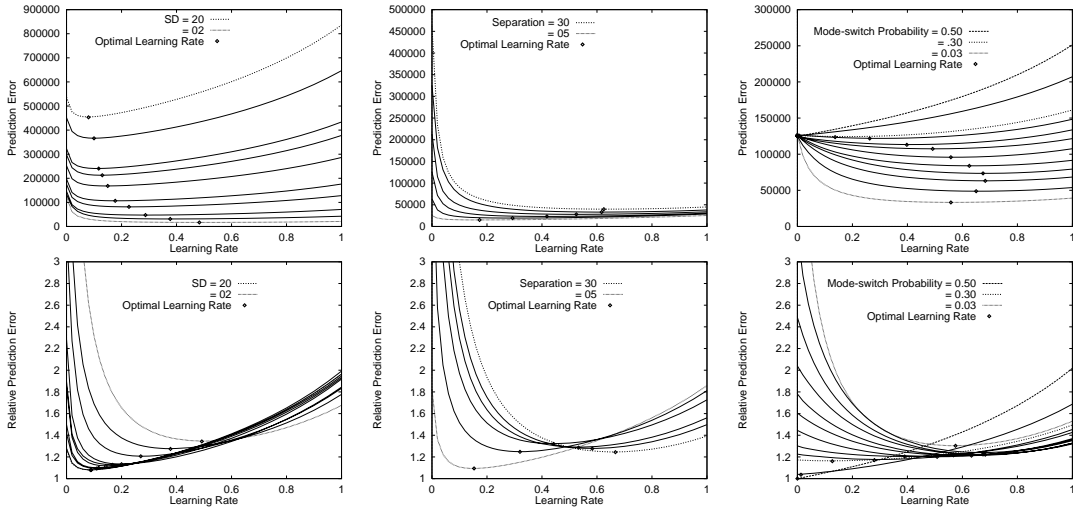


Figure 2. Error versus learning rate, varying variance, separation between mode means, and probability of mode switches. First row gives absolute error, second row gives error relative to the performance of the Bayesian model-based algorithm.

	μ_1	σ_1	λ_{12}	μ_2	σ_2	λ_{21}	error for fixed grid	error for best fixed α
1.	10	2	0.01	20	2	0.01	1.3095	1.3132
2.	10	2	0.1	20	2	0.1	1.2529	1.2492
3.	10	4	0.01	20	4	0.01	1.1269	1.1259
4.	10	4	0.1	20	4	0.1	1.1551	1.1529
5.	10	4	0.01	100	4	0.01	1.1394	1.1366
6.	10	4	0.1	100	4	0.1	1.1097	1.1091
7.	10	1	0.01	30	8	0.01	1.2120	1.2075
8.	10	1	0.01	30	8	0.02	1.2646	1.2620

Table 1. Results on a set of test cases

current sample, it is also told the mode that the sample came from. Thus, although the algorithm doesn't know what mode the next sample will come from, it knows with certainty the mode of the previous sample.

To make its guess, the algorithm takes the mode i of the previous sample, then predicts $\hat{x}_i = \sum_j \lambda_{ij} \mu_j$. The logic for this parallels that of the Bayesian model-based algorithm in Section 4.1. It is selecting the mean of the distribution from which the next sample will be drawn.

The error of the optimal algorithm can't be any lower than this algorithm since it optimally uses information that is not available to any real prediction algorithm.

What is the error of this insider algorithm? Because it always knows the previous mode with certainty, we only need to consider how it behaves over consecutive transitions. In particular, for any pair of modes i, j , the probability that a randomly chosen transition will be from i to j is $\pi_i \lambda_{ij}$, where π is the stationary distribution of the Markov chain. The expected error for such a transition will be $E[(X_j - \hat{x}_i)^2] = \sigma_j^2 + \mu_j^2 - 2\mu_j \hat{x}_i + \hat{x}_i^2$.

For a 2-mode environment, the expected error for the cheating algorithm is

$$\frac{\lambda_{21}}{\lambda_{21} + \lambda_{12}} \sigma_1^2 + \frac{\lambda_{12}}{\lambda_{21} + \lambda_{12}} \sigma_2^2 + \frac{\lambda_{21} \lambda_{12}}{\lambda_{21} + \lambda_{12}} (\mu_1 - \mu_2)^2 (2 - \lambda_{21} - \lambda_{12}). \quad (2)$$

Because we assume that self-transitions occur with probability at least one half, we have that $0 \leq \lambda_{21} + \lambda_{12} \leq 1$.

How does this compare to the error for the fixed learning-rate approach using $\alpha = 1$? Once again, the structure of the algorithm (always predict the previous value) means we only need to consider how it behaves over consecutive transitions (i to j). The expected error for such a transition will be $E[(X_j - X_i)^2] = \sigma_j^2 + \mu_j^2 - 2\mu_j \mu_i + \sigma_i^2 + \mu_i^2$.

For a 2-mode environment, the expected error for the $\alpha = 1$ algorithm is

$$\frac{2\lambda_{21}}{\lambda_{21} + \lambda_{12}} \sigma_1^2 + \frac{2\lambda_{12}}{\lambda_{21} + \lambda_{12}} \sigma_2^2 + \frac{2\lambda_{21} \lambda_{12}}{\lambda_{21} + \lambda_{12}} (\mu_1 - \mu_2)^2. \quad (3)$$

Given our constraints on λ , the ratio of Equation 3 to Equation 2 can be no more than two. Therefore, the ratio between the error using the best fixed setting for α and the optimal learning algorithm is bounded by two. There are two-mode environments, such as the one illustrated in the bottom right of Figure 2 (mode switch probability at 1/2), in which the performance of the algorithm actually matches this bound.

As a bound on the relative performance of the optimal α , the bound appears loose. In our simulations, the ratio between the error for the best setting of α and that of the Bayesian model-based algorithm seems to be between 1.1 and 1.4. So, if we know the right α to use, we can get fairly close to optimal performance without learning a model.

8. Conclusion

This paper introduced a non-stationary prediction problem in which environments are modeled by hidden Markov models with real-valued observations and highly likely self transitions. We argued that this is a class of environments with many real-world applications. We showed that a simple algorithm based on learning with a fixed learning rate can do well in this class of environments relative to optimal performance. We proved that we can set the learning rate to guarantee performance within a factor of 2 of optimal, and demonstrated a grid-based learning algorithm that performs even better on a set of simulated examples.

We plan to explore extensions of this work including using an adaptive or variable-resolution grid and using the algorithm for value prediction in multi-state hidden mode Markov decision processes.

Acknowledgments

Thanks to Satinder Singh Baveja, Peter Stone, Sam Choi, Sanjoy Dasgupta, Michail Lagoudakis, Charles Isbell and others for helpful advice during the course of this project.

References

- Choi, S. P., Yeung, D.-Y., & Zhang, N. L. (2000). An environment model for nonstationary reinforcement learning. *Advances in Neural Information Processing Systems*. The MIT Press.
- Dayan, P., & Sejnowski, T. J. (1996). Exploration bonuses and dual control. *Machine Learning*, 25, 5–22.
- Littman, M. L., & Ackley, D. H. (1991). Adaptation in constant utility non-stationary environments. *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 136–142). San Mateo, CA: Morgan Kaufmann.
- Stone, P. (2000). TPOT-RL applied to network routing. *Proceedings of the Seventeenth International Conference on Machine Learning*. San Francisco: Morgan Kaufmann.

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning.
Machine Learning, 8, 279–292.