Extended Markov Tracking with an Application to Control

Zinovi Rabinovich and Jeffrey S. Rosenschein
School of Engineering and Computer Science
The Hebrew University of Jerusalem
91904 Jerusalem, Israel
{nomad, jeff}@cs.huji.ac.il

Abstract

We present here a tracking method for Markovian system dynamics which operates in an on-line fashion, during interaction with the system. Existence of a simple numerical solution of the technique is demonstrated. We also demonstrate the application of the tracking method in a novel continual planning architecture, together with experimental data to support the validity of the presented tracking approach.

1. Introduction

Imagine an artificial intelligence (AI) system capable of serving as a car's autopilot. Using satellite positioning and maps, the car navigates through city streets, moving from a starting location to some defined end location. Notice, however, that there are really two separate kinds of tasks being carried out by the AI autopilot. The "high-level" task decides, based on constraints like street connectivity and one-way traffic restrictions, how to go from point A to point B. The "low-level" task is to make sure that the car maintains other constraints as it progresses, that it follows the curve of the road, adjusts the car's steering in response to dynamic feedback from the environment, and applies brakes as necessary. It is the *interaction* of these two levels that results in successful completion of the task.

Besides reasoning about the immediate tasks at hand, two activities are of major importance: environment evolution (a.k.a. dynamics) tracking, and failure detection.[1] Environment-dynamics tracking and failure detection are the "supports" that keep any continual planning scheme together.

1 In our car scenario, this might be keeping track of the streets the car has already passed, as well as detecting the situation where the path becomes accidentally blocked.

Unfortunately, most tracking and detection techniques attempt to look into the reasons behind events, thus creating complex dynamic models that require a large amount of computation and observational data [3, 11, 10]. Although such models eventually provide accurate and insightful images of the world, in many cases we are simply not interested in the internal wiring of the system, but need, rather, a simple surface explanation of its ongoing interaction with the environment.

Pursuing this demand for simplicity and the on-line exploitation of minimal data, we develop a Markov-dynamics tracking mechanism, which operates during the system's interaction with the environment as described in Section 2. To demonstrate the effectiveness of our approach, we have integrated it into a continual planning architecture (Section 3), and set up an experimental domain as discussed in Section 3.2.

2. Extended Markov tracking

Determining, from observations, the behavioral trends of an environment during interaction is known to be a hard problem. It is universally treated by complex models that require significant amounts of data and involve a large computational burden [11, 3, 10]. However, most of the time interaction is either short-lived or, although internally complex, exhibits simple external trends. Thus, a simple Markovian model will often be sufficient for our needs.

2.1. Model

A Markovian environment is described by a tuple $< S, A, T, O, \Omega, s_0 >$, where:

- S is the set of all possible environment states;
- s_0 is the initial state of the environment (which can also be viewed as a distribution over S);
- A is the set of all possible actions applicable in the environment;

- T is the environment's probabilistic transition function: $T : S \times A \rightarrow \Pi(S)$. That is, $T(s'|a, s)$ is the probability that the environment will move from state s to state s' under action a;
- O is the set of all possible observations. This is what the sensor input would look like for an outside observer;
- Ω is the observation probability function: $\Omega : S \times A \times S \rightarrow \Pi(O)$. That is, $\Omega(o|s', a, s)$ is the probability that one will observe o given that the environment has moved from state s to state s' under action a.

In current approaches, "tracking" would mean a continual estimation of the system state as seen through the distortion of observations. We find this approach insufficient, since it disregards the purpose of continual tracking: rather than understanding *where* the system is moving, it is of greater importance to know *how* it is moving. In other words, instead of tracking system state, we need to track the way the system changes, i.e., the dynamics of the system.

The original reason for tracking system state was due to a set of preferences over the state; in dynamic systems, however, there exists an additional preference, a preference over system transitions. This preference is basically a demand, or limitation, on the exhibited system state dynamics. The *way* a system changes is the target of the tracking approach proposed here. We would like to know what kind of behavior the system exhibits, that is, under a Markovian environment description, we would like to keep an estimate $PD : S \rightarrow \Pi(S)$.

2.2. Tracking and Evaluation

We propose to track the system's exhibited dynamics by continual updating of the model, subject to keeping a conservative distance between old and new models. In other words, given our previous system dynamics estimate PD, we seek a new estimate D that can account for the observation at hand, while staying as close as possible to PD.

The distance between old and new system estimates is performed using the Kullback-Liebler (KL) divergence function [5]:

$$D_{KL}(p\|q) = \sum_x p(x) \log \frac{p(x)}{q(x)}.$$

From the information theory point of view, this is the price one will have to pay for using distribution q to encode a source distributed by p. In our case, the old estimate is the encoding we use, while the new estimate represents the true source distribution. In a sense, we measure the "regret" of using an old estimate in light of new evidence and a new estimate of the system dynamics. Since we seek a conservative update, we would like the regret to be minimized.

However, to complete the measure and system dynamics tracking, one has to maintain one auxiliary kind of information: beliefs about the current system state. Initial beliefs are given by the definition in the environment tuple. Assume that at some stage we believe $p \in \Pi(S)$, and we obtain an observation $o \in O$ after action a. Then we can update our beliefs using the Bayesian formula:

$$\tilde{p}(s) \propto \Omega(o|s, a) \sum_{s'} T(s|a, s')p(s').$$

Notice that, although action is present in computations of the state estimator, it serves as an input to the procedure, and not as an unknown parametric component.

Given that we keep track of our beliefs about the environment state, we can do the same with regard to the observed dynamics. Our prior beliefs in this case might obviously depend on the domain, but in the absence of any other preference, one commonly accepted prior belief is the uniform distribution. That is, we assume at the beginning that anything can happen in the environment with equal likelihood. The other popular alternative is a 'static' environment, that is, the assumption that the system state does not change.

When we update our beliefs about the system dynamics $PD(s'|s)$ (the observed system dynamics), we would like the new version of it, $D(s'|s)$, to explain what happened in the system from our point of view: $\tilde{p}(s') = \sum_s D(s'|s)p(s)$, where $p(\cdot)$ and $\tilde{p}(\cdot)$ are, respectively, the old and the new estimations of the system state. Obviously, though, we do not want to wander too far from our current beliefs about system dynamics. Thus the update is a solution to the following optimization problem:

$$D(s'|s) = \arg \min_{Q(s'|s)} \langle D_{KL}(Q(s'|s)\|PD(s'|s)) \rangle_{p(s)}$$
$$s.t.$$
$$\forall s' \;\; \tilde{p}(s') = \sum_s Q(s'|s)p(s)$$
$$\forall s \;\; \sum_{s'} Q(s'|s) = 1$$

Notice that one explanation is trivial: it is the matrix $A_0 = (\tilde{p}, ..., \tilde{p})$, composed of the newly observed system state distribution for columns. However it is not optimal, relative to our prior system dynamics PD. Fortunately, the set of all feasible explanations for the change of system state distribution forms a convex hull. It is possible to construct a linear base for the space of neutral matrices: $\{O_t \in M_{n\times n}(\mathcal{R})|t \in T, \vec{1} * O = \vec{0}, O * p = \vec{0}\}$, thus expressing D as a linear sum: $D = A_0 + \sum_{t\in T} \alpha_t O_t$, where $\{\alpha_t\}_{t\in T}$ are some real coefficients. This allows us to refor-

mulate the optimization problem as:

$$D = \arg \min_{\{\alpha_t\}_{t \in T} \subseteq \mathcal{R}^{|T|}} \left\langle D_{KL}(A_0 + \sum_{t \in T} \alpha_t O_t \| PD) \right\rangle_{p(s)}$$
$$s.t.$$
$$A_0 + \sum_{t \in T} \alpha_t O_t > 0$$

It is possible to explicitly construct the $\{O_t\}$ base as the tensor product of sets of vectors perpendicular to $\vec{1}$ and $\vec{p}$. Although D_{KL} is not a true metric over the space of probability distributions, its convexity and the linearity of constraints over α_t allow the problem to be efficiently solved using relatively standard numerical optimization techniques, e.g., gradient descent.

Notice that the Kullback-Leibler extension for Markovian dynamics $\langle D_{KL}(Q(s'|s) \| PD(s'|s)) \rangle_{p(s)}$ can also provide *failure detection* for controllers. This was used in a Strategic/Tactical planning architecture that we have developed (described below), providing empirical support for the validity of this tracking approach.

3. An Application to Tactical Planning

The Strategic/Tactical framework of planning consists of two continually interacting layers:

- The *strategic* layer of the planner is concerned with the following question: given a high level goal, what kind of system dynamics will suffice to achieve it;
- The *tactical* layer of the planner has no concern whatsoever with the high level goal. Rather, it attempts to ensure that the system dynamics of the changing state indeed match the one desired by the strategic layer.

Failure of the tactical layer is reported back to the strategic layer, forming a continuous loop of plan enhancement.

Although the Strategic/Tactical paradigm of planning is evident in many old and new planners [7, 1, 14, 9], we believe it is necessary to shift attention to the abilities and responsibilities of the tactical layer, which typically is present only in a degenerate sense within existing planners.

To enhance the Tactical layer, we used extended Markov tracking as presented above (Section 2) with a novel notion of the planning goal. Instead of using the classical POMDP approach (reward function and expected accumulated reward) to guide action selection, we made action selection be directly dependent on system dynamics. The Strategic layer thus provides a *tactical target*, a distribution (or a preference function) $r : S \to \Pi(S)$, while the Tactical layer (through system-dynamics tracking) has to ensure wise action selection.

Denote by $H(\bar{p}, p, PD)$ the procedure of obtaining the optimal explanation for transition between belief states p and $\bar{p}$ with respect to the dynamics-prior PD. Then, the action of choice in our extended Tactical layer is:

$$a^* = \arg\min_a \langle D_{KL}\left(H(T_a p, p, PD) \| r\right) \rangle_p$$

where T_a is the environment transition function restricted to action a ($T_a p$ thus becomes a matrix applied on a vector).

The overall tactical algorithm may be written as follows:

0. Initialize estimators:
   - the system state estimator $p_0(s) = s_0 \in \Pi(S)$,
   - system dynamics estimator
   $$PD_0(\bar{s}|s) = prior(\bar{s}|s)$$
   Set time to $t = 0$.
1. Select action a^* to apply using the following computation:
   - For each action $a \in A$ predict the future state distribution $\bar{p}^a_{t+1} = T_a * p_t$;
   - For each action, compute
   $$D_a = H(\bar{p}^a_{t+1}, p_t, PD_t)$$
   - Select $a^* = \arg\min_a \langle D_{KL}\left(D_a \| r\right) \rangle_{p_t}$
2. Apply the selected action a^* and receive an observation $o \in O$.
3. Compute p_{t+1} due to the Bayesian update.
4. Compute $PD_{t+1} = H(p_{t+1}, p_t, PD_t)$.
5. Set $t := t + 1$, goto 1.

In essence, the tactical algorithm above utilizes extended Markov tracking both to guide its action selection, and to ensure that the exhibited behavior indeed concurs with the given preference $r : S \to \Pi(S)$.

3.1. Evaluation

To see how the system tracking and the overall tactical layer performed, we designed a simple experiment, presented below in Section 3.2. First, however, consider the evaluation criteria that are worthwhile to apply on the experimental data. Since the system in question is stochastic by nature, a probabilistic evaluation of performance has to be applied. The following criteria were used to compare different tactical solutions:

Definition 1 *Let Y^n_τ denote the set of all possible values of divergence from the tactical target obtained by tactics τ during the first n steps. Define a probability distribution of the divergence due to Y^n_τ as*

$$P_\tau(a < x < b) = \frac{\#\{y \in Y^n_\tau | a < y < b\}}{n}.$$

Definition 2 *For two different tactics τ_1 and τ_2 and a threshold value ξ, we say that τ_1 outperforms τ_2 (in n steps) if $P_{\tau_1}(x < \xi) > P_{\tau_2}(x < \xi)$. That is, one is less likely to break some divergence limit ξ using tactics τ_1, than using tactics τ_2.*

Definition 3 *For two tactics τ_1 and τ_2, denote D_1 and D_2 as the probability distributions of the divergence due to τ_1 and τ_2, respectively. We say that τ_1 definitely outperforms τ_2 if $P(D_1 < D_2) > \frac{1}{2}$.*

As one support for this approach to comparison, consider [6], where the same technique was used to balance exploration and exploitation in a stochastic domain.

The second question regarding evaluation of our tracking technique concerns the distance measure between observed and required dynamics. Optimization applied during system tracking rules out the extended Kullback-Leibler divergence from being used during evaluation. Instead, we view system dynamics as linear transformations, and measure cumulative distance between image vectors:

$$\int_{\Omega} \|A\omega - B\omega\|^2 \partial\Omega.$$

In addition, a domain-specific measure of success was applied in our experiment set, to demonstrate the validity of comparison.

3.2. Experimental Data

Since no truly tactical alternative solution currently exists, we used a standard POMDP solver [4] as a point of reference for our technique.

Experiments were carried out in the Drunk Man Walk domain (among others); see Figure 1. This is a classic example for Markov chain discussions: a man stumbles along a path between his home and nettle bushes, making random steps left (home) and right (away from home).

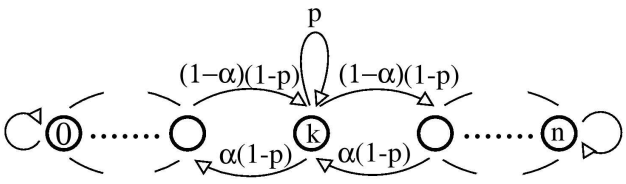

Figure 1. Drunk Man walk state diagram. Parameter α is subject to action effect. p is the stabilization parameter.

The setup was modified to allow stochastic control—actions can be applied to tilt the probability balance between left and right steps, but the number of steps is non-trivial (in our experiments 1/4 of the path could be passed in a single epoch). In addition, the true position of the man was veiled by the observation set—which is the same size as the set of possible positions. The observation probabilities essentially just "blur" the system state. The task was to keep the man as far away from home and nettle bushes alike. The classic delayed reward schema, where one is rewarded if the man reached the middle of the path, was translated into a transition preference matrix for our tactical layer. The comparison was done based on the distribution of distance from the path midpoint.

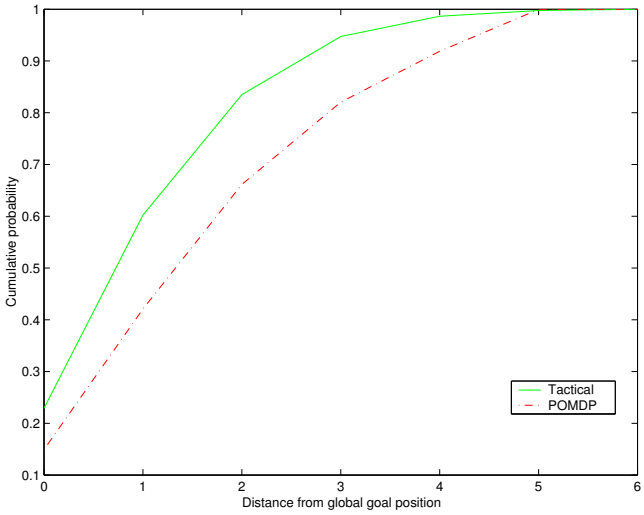

Figure 2. Divergence from Global Goal (midway point)

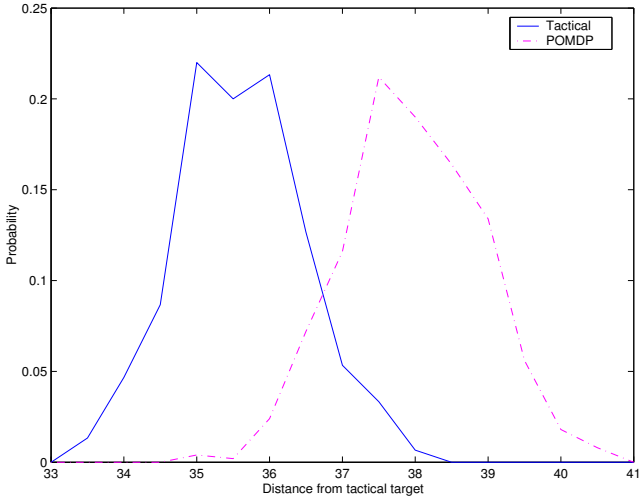

Figure 3. Tactical Distance Divergence

The exact setting is as follows:

- Denote by n the number of steps between home and nettle.
- Denote the number of possible directional inclinations by F^k_{num}; together with one neutral action it forms a set of $F_{num} = 2 * F^k_{num} + 1$ possible actions.
- Denote by p_0 the initial predisposition of the drunk man to stay put.
- Denote by ξ the number of probabilistic steps taken by the drunk man under the influence of a single action application.
- Then the probability vector of left, zero, and right steps respectively (p^-, p, p^+) is computed for each action $1 \leq i \leq F_{num}$, using the following formulas:

$$\begin{aligned} F &= (i - F^k_{num} - 1)/F^k_{num} \\ p &= p_0 \frac{1 - |F|}{2} \\ \alpha &= \frac{F + 1}{2} \\ p^- &= \alpha * (1 - p) \\ p^+ &= (1 - \alpha) * (1 - p) \end{aligned}$$

In the experiment set, which resulted in the data shown in Figure 2 and Figure 3, the parameters where set to: $F^k_{num} = 5, n = 12, p_0 = \frac{1}{5}$, and $\xi = 3$. Preference over the state transitions was set to $P(\bar{x}|x) = \begin{cases} \frac{1+\epsilon}{Z(x)} & \bar{x} = \lfloor \frac{n+1}{2} \rfloor \\ \frac{\epsilon}{Z(x)} & otherwise \end{cases}$, where $Z(x)$ is a normalization factor. The classical POMDP solution used the same preference as a reward function, and attempted to accumulate as large a reward as possible over a preset number of steps. The observation set was equally large as the system state set, and probabilities were set so that observation was equally distributed over the immediate neighborhood of the man's true position.

The tactical layer *definitely outperformed* the classical POMDP solution. Figure 3 shows statistics of the tactical and classical POMDP solutions with respect to the tactical distance. It can easily be seen that the divergence of the tactical solution from the required transition preference will be, with high probability, less than the divergence exhibited by the classical POMDP solution. However, since tactical distance is a new concept, the success of the tactical solution was verified using another measure. Figure 2 shows the cumulative probability that the Drunk Man will be away from the midpoint along the path. It can easily be seen that the proposed tactical solution once again is more likely to be close to the midpoint (distance zero) than the classical POMDP solution. It is also important to note that the tactical layer in these experiments was applied in an on-line fashion, while the POMDP solution was obtained offline with prior knowledge of the problem horizon (which improved its performance).

4. Related work

Although the choice above in Section 2 of the Kullback-Liebler distance may seem arbitrary, it is inspired by work in information theory [12, 13] and information gathering [2, 8].

Tishby et al. [12, 13] proposed a novel method for distributional clustering and relevant quantization. In [13], Tishby, Pereira and Bialek developed an elegant theoretical schema for preserving maximum relevant information, when it has to be transmitted via a third party. For example, image-based face recognition has two stages: image "compression" into features, then face recognition based on the features. [13] provides an exact, self-consistent set of equations that allow optimal creation of feature coding, so that information relevant to face recognition will be maximally preserved in the derived features.

Later work [12] provides a practical algorithm for developing the feature set itself, also through a widely applicable mathematical construction. In our work, we used similar mathematical tools. However, there is an important difference in the focus of our approach. Tishby's team was interested in the optimization of coding for some given environment, while our work deals with the inverse concept—change the world to fit some preset coding (our action capabilities and target).

An information-theoretic approach to control was also used by the research group of Durrant-Whyte [2, 8]. In this work, multi-robot sensor systems are discussed and put into the framework of decentralized data fusion. As a consequence, control becomes a function of predicting information gain from an action, and choosing the best prospect. Allowing non-frequent exchange of observation information, a successful control schema is obtained. Having said that, it is important to notice that the goal of their systems is information itself. In our work, we tend toward more general systems, where information is only a key to better control, not the ultimate goal.

5. Conclusions and Future work

We have introduced extended Markov tracking of system dynamics, along with divergence and distance measures that provide a basis for plan failure detection. The on-line nature of the tracking mechanism, and its connection with system interaction, allowed us to integrate it into a hierarchal continual planning architecture.

In spite of the fact that belief update may generally be hard to perform, for discrete systems the proposed method

is completely tractable. Moreover, extended Markov tracking does not introduce any substantial additional computational complexity, even in the continuous case, since the main computational leverage comes from system state update, already present in most tracking systems.

Although a formal proof of efficiency is not yet available, experimental data supports the validity of the extended Markov-tracking technique presented here. It also demonstrates that system-dynamics tracking can significantly improve continual planning performance.

We plan to expand our experimental investigation and to introduce the technique into multiagent domains. In addition, further investigation of the tracking technique's theoretical and on-line properties, such as its competitiveness and stability, are among the many points of analysis that remain for future work.

References

[1] J. Blythe. *Planning under uncertainty in dynamic domains*. PhD thesis, CMU, CS Department, 1998.

[2] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, and H. Durrant-Whyte. Information based adaptive robotic exploration. In *IROS*, 2002.

[3] H. H. Bui. A general model for online probabilistic plan recognition. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 1309–1315, 2003.

[4] A. R. Cassandra. POMDP solver software. http://www.cassandra.org/pomdp/code/index.shtml, 2003.

[5] T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley, 1991.

[6] R. Dearden, N. Friedman, and S. Russel. Bayesian Q-learning. In *Proceedings of AAAI*, pages 761–768, 1998.

[7] R. E. Fikes and N. J. Nilsson. STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[8] B. Grocholsky, A. Makarenko, and H. Durrant-Whyte. Information-theoretic coordinated control of multiple sensor platforms. In *ICRA*, 2003.

[9] D. S. Nau, S. J. J. Smith, and K. Erol. Control strategies in HTN planning: Theory versus practice. In *AAAI/IAAI*, pages 1127–1133, 1998.

[10] D. V. Pynadath and M. P. Wellman. Probabilistic state-dependent grammars for plan recognition. In *Proceedings of the 16th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 507–514, 2000.

[11] P. Riley and M. Veloso. Recognizing probabilistic opponent movement models. In A. Birk, S. Coradeschi, and S. Tadokoro, editors, *RoboCup-2001: The Fifth RoboCup Competitions and Conferences*. Springer Verlag, 2002.

[12] N. Slonim and N. Tishby. Agglomerative information bottlenecks. In *Proceeding of NIPS-12*, pages 617–623. MIT Press, 2000.

[13] N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. In *The 37th annual Allerton Conference on Computation, Control and Computing*, pages 368–377, 1999.

[14] M. Veloso, J. Carbonell, A. Prez, D. Borrajo, E. Fink, and J. Blythe. Integrating planning and learning: The PRODIGY architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, 7(1):81–120, 1995.

[15] R. Washington. Markov tracking for agent coordination. In *Agents-98: The Second International Conference on Autonomous Agents*, pages 70–77, 1998.