2012 IEEE International Conference on Systems, Man, and Cybernetics
October 14-17, 2012, COEX, Seoul, Korea

Acquiring a Broad Range of Empirical Knowledge in Real Time by Temporal-Difference Learning

Joseph Modayil, Adam White, Patrick M. Pilarski, Richard S. Sutton
Reinforcement Learning and Artificial Intelligence Laboratory,
Department of Computing Science, University of Alberta, Canada

***Abstract*—Several robot capabilities rely on predictions about the temporally extended consequences of a robot's behaviour. We describe how a robot can both learn and make many such predictions in real time using a standard algorithm. Our experiments show that a mobile robot can learn and make thousands of accurate predictions at 10 Hz. The predictions are about the future of all of the robot's sensors and many internal state variables at multiple time-scales. All the predictions share a single set of features and learning parameters. We demonstrate the generality of this method with an application to a different platform, a robot arm operating at 50 Hz. Here, learned predictions can be used to measurably improve the user interface. The temporally extended predictions learned in real time by this method constitute a basic form of knowledge about the dynamics of the robot's interaction with the environment. We also show how this method can be extended to express more general forms of knowledge.**

I. Introduction

Predicting the temporally extended consequences of behaviour in real time provides a foundation for many robot capabilities. Examples include collision avoidance (Fox et al., 1997), model predictive control for stability (Abbeel et al., 2010), and motion planning (LaValle, 2006). The conventional approach to make these predictions is to manually construct a small one-timestep model of the system dynamics offline, and then, during real-time operation, to make temporally extended predictions by simulating future trajectories with the model. However, this approach requires a one-timestep model of the dynamics to be available, and it requires computationally expensive simulations with the model to predict quantities of interest.

We propose an alternative approach for real-time predictions, namely to learn to directly predict the temporally extended consequences of a behaviour. This is the same technique used with the critic's value function in an actor-critic based method. We demonstrate that this direct approach scales well, and is a viable method for learning and making many temporally extended predictions in parallel.

The main contribution of this work is an empirical demonstration that thousands of temporally extended predictions can be learned online in real time with high accuracy. We demonstrate that a mobile robot can both make and learn thousands of predictions in real time. Predictions are made every 100ms, and the predictions are about the robot's future sensor readings and internal state variables either at the next timestep in 100ms, or over the next short time scale of 0.5, 2, or 8 seconds. These predictions provide the robot with immediate knowledge about many distinct, temporally extended consequences of its behaviour. In a second experimental setting, we demonstrate the generality of these predictions by evaluating how they can improve the user interface for a robot arm.

The approach is novel in several respects. The predictions have the benefit of scientific empiricism—the predictions can be evaluated for their accuracy by comparison to the robot's future experience. Although directly learning the temporally extended consequences of behaviour is not a common way of representing knowledge in robotics, these predictions can also be assembled to form a conventional one-timestep model of the dynamics. The ease of acquiring knowledge, the generality of the method, and known extensions to the prediction algorithm, suggest this is a promising direction for further investigation.

The paper is structured as follows. First, we describe the learning setting and present our method. Then, we show results from our experimental evaluation of the method on a mobile robot. We demonstrate the generality of this method with an application to the completely different domain of predictions for a human-guided robot arm. After describing related work, we discuss how this method can be extended to more general forms of prediction.

II. Method

The method relies on learning many temporally extended predictions, so we first review the underlying temporal-difference prediction algorithm TD(λ) (Sutton and Barto, 1998). As input at each timestep $t \in \mathbb{N}$, the algorithm receives the feature vector $x_t \in \mathbb{R}^n$ (typically binary-valued in practice). The feature vector is the robot's description of the state of the environment s_t. Note that the description provided by x_t will be restricted to features that the robot can readily compute, and this is typically an incomplete characterization of the state of the environment. Each predictive question pertains to some signal $r_t \in \mathbb{R}$ that is observed at each timestep. The signal is called the reward in reinforcement learning, but here it is an arbitrary target signal and does not indicate a quantity that the robot wishes to maximize. We assume that the robot is following a fixed behaviour and the question is to predict the *return*, G_t, which is the discounted sum of the target signal

This work was supported by grants from Alberta Innovates - Technology Futures, the National Science and Engineering Research Council of Canada, the Alberta Innovates Centre for Machine Learning, and the Glenrose Rehabilitation Hospital Foundation.

978-1-4673-1714-6/12/$31.00 ©2012 IEEE

observed in the future,

$$G_t = \sum_{k=1}^{\infty} \gamma^{k-1} r_{t+k}, \tag{1}$$

where γ is a constant. A particular choice of γ will focus the question on either the next timestep (note $\gamma = 0$ implies $G_t = r_{t+1}$) or over an extended temporal horizon for $\gamma \in (0, 1)$. The linear TD(λ) algorithm learns to approximate the expected return by a linear function of the feature vector x_t, with

$$\hat{g}(x_t) = \theta_t^\top x_t,$$

where $\theta_t \in \mathbb{R}^n$. Prediction is computationally efficient in that the time and space requirements are linear in the feature vector size. The TD(λ) algorithm adjusts the weight vector θ_t at each timestep to reduce the error between predictions on adjacent timesteps with the following update rules.

$$\delta_t = r_{t+1} + \gamma \theta_t^\top x_{t+1} - \theta_t^\top x_t \tag{2}$$

$$e_t = \gamma \lambda e_{t-1} + x_t \tag{3}$$

$$\theta_{t+1} = \theta_t + \alpha \delta_t e_t \tag{4}$$

Here $e_t \in \mathbb{R}^n$ is called the trace (and is initialized to the zero vector), and $\alpha \in [0, 1)$ is a step size parameter. The value $\lambda \in [0, 1]$ is the trace decay parameter. When $\lambda = 1$ and α is slowly decreased over time to zero, this algorithm converges to a weight vector θ_* that minimizes the squared error between the predictions and the return. However, the algorithm is often used with $\lambda < 1$ for faster learning, and with α set to a constant value to enable adaptation to a dynamic environment. Note that the update at each timestep requires time that is linear in the feature vector size.

Under common assumptions (Sutton and Barto, 1998), the update rules will adapt $\hat{g}$ to approximate g, a general value function that is the expected value of the return when starting at the environmental state s.

$$\hat{g}(x(s)) \approx g(s) \equiv E[G_t | S_t = s]$$

A common oversight is to consider TD(λ) as only appropriate for learning a value function that describes the robot's behaviour. It is in fact a general algorithm for making multistep predictions, and was described as such when introduced (Sutton, 1988). Although this algorithm is often used in reinforcement learning to pursue goal-directed behaviour, it can be used for an arbitrary function r_t of state.

We propose taking advantage of the computational efficiency of the TD(λ) algorithm to learn a set of m predictions,

$$\{(r^{(1)}, \gamma^{(1)}), \ldots, (r^{(m)}, \gamma^{(m)})\}$$

that can be learned and predicted in parallel from the single stream of robot behaviour. Each predictive question has its own target signal r and constant γ. As the learning and prediction algorithms are both linear in the n-dimensional feature vector, the computational complexity and memory requirements of this approach grow as $O(mn)$, which on modern computing systems enables the use of many features and many predictions in real time. Moreover, this approach is intrinsically parallel and decoupled, which enables flexible deployment on parallel computing architectures.

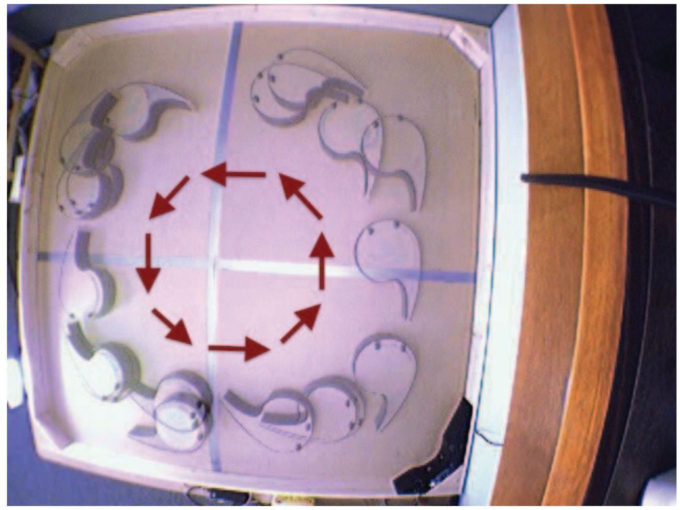

Fig. 1. A robot performing a regular though non-periodic behaviour of following the walls in the pen. A lamp shines in one corner of the pen, and its light is observed by some of the robot's sensors.

The goal of learning to make many predictions in parallel in real time on a robot raises different considerations than are often considered for reinforcement learning experiments. In particular, manually tuning the learning parameters for each question is impractical. Instead, the learning parameters should enable stable learning. As such, values for α and λ are shared across all the questions. Furthermore, the feature vector is shared across all the questions. This problem setting encourages the use of diverse features and a large feature vector, to enable learning better predictions for a broad set of questions. Note that in the online setting with an abundance of data, increasing the space of features is generally not harmful.

We define *in situ* learning as this scenario of learning and predicting in real time on a robot. Two competing desires for *in situ* learning must be balanced. First, learning and prediction are often only one piece of the larger system, so their computational and memory footprints should be reasonable. Second, the predictions should exhibit accuracy within the robot's lifetime. Several approaches in robot learning are computationally expensive but learn from limited experience. However, modern robots have extended operational lifespans of days and years, so computationally efficient real-time algorithms can run on top of these operational systems with little overhead, and thus enable learning to occur directly from the stream of robot experience.

III. Evaluation

To evaluate the practicality of the above method on a real system we considered *nexting* predictions, namely predictions about the future value of sensors (and many feature vector components), at a variety of time scales (as described originally by Modayil et al. (2012)). By predicting what

Sensor Group	Group Size	Tiling Type	(resolution, tilings)
IRDistance	10	strip	(8,8)
		strip	(2,4)
		skip(0)	(4,4)
		skip(1)	(4,4)
Light	4	strip	(4,8)
		skip(0)	(4,1)
IRLight	8	strip	(8,6)
		strip	(4,1)
		skip(0)	(8,1)
		skip(1)	(8,1)
Thermal	8	strip	(8,4)
Rotational Velocity	1	strip	(8,8)
Mag	3	strip	(8,8)
Accel	3	strip	(8,8)
MotorSpeed	3	strip	(8,4)
		skip(0)	(8,8)
MotorVoltage	3	strip	(8,2)
MotorCurrent	3	strip	(8,2)
MotorTemperature	3	strip	(4,4)
OverheatingFlag	1	strip	(2,4)
LastAction	3	strip	(6,4)

Fig. 2. Summary of the tile-coding strategy for producing the feature vector from the sensory observations. Sensors values in each group were tiled either singly (strip tilings) or jointly pairwise (skip tilings). The last column indicates how many tilings of each type were made for each sensor or sensor group, and how many intervals (resolution) were involved in each dimension of each tiling. See text for explanation.

will happen next, the robot gained a basic knowledge of its interaction with the environment. In our experiment, the robot performed an extended wall-following behaviour in a small pen (Fig. 1). The observation stream contained both repeated events (such as passing a light, driving forward, and periods when the motors are cooling off), along with fine structure (such as variations in the accelerometers without readily apparent structure). The behaviour exhibited substantial variations, for example the time to complete a loop of the pen varied from 20 to 40 seconds, and there were intermittent seven-minute resting periods with no motion to allow the motors to cool off. Every 100ms, the robot generated an observation vector with 53 components. They cover 11 different sensing modalities and 4 software variables that are listed in Fig. 2.

The observation vector is transformed into the agent's representation x_t by tile coding. This produced a binary vector, $x_t \in \{0,1\}^n$, with a constant number of 1 features (see Sutton and Barto (1998) for more details; in short a tile coder maps data from a continuous domain into a binary representation by a set of indicator functions whose support tile the continuous domain). The features provided no history and performed no averaging of sensor values. The tile coder was comprised of many overlapping tilings of individual sensors and pairs of sensors (see Fig. 2). The *resolution* of a tiling refers to the number of uniform partitions per dimension. When multiple tilings covered a space, each had a random offset. The sensory signals were partitioned based on sensor modalities into IR(InfraRed)Distance, Light, Thermal, IRLight, MotorSpeed, MotorCurrent, MotorVoltage, MotorTemperature, Acceleration, Magnetometer and LastAction. Within each sensor group, each individual sensor (e.g., Light0) was tiled independently as multiple one-dimensional overlapping grids called *strip* tilings. Additionally, pairs of sensors within a group (e.g., IRLight$i$ and IRLight$j$) were tiled together using multiple two-dimensional overlapping grids. The two-dimensional grids combined sensors in one of two ways. When they combined sensors within a group that were directly spatially adjacent on the robot, we call it a *skip(0)* tiling, whereas a *skip(1)* tiling combines sensors that are spatially adjacent with a skip of one (e.g., IRDistance1 with IRDistance3, IRDistance2 with IRDistance4, etc.). All in all, this tiling scheme produced a feature vector with $n = 6065$ components, most of which were 0s, but exactly 457 of which were 1s, including one bias feature that was always 1.

We applied TD(λ) to learn 2160 predictions in parallel. For the first 212 predictions, the target signal, $r^{(i)}$, was the sensor reading of one of the 53 sensors listed in Fig. 2 and the discount rate, $\gamma^{(i)}$, was set to one of four timescales; for the remaining 1948 predictions, the target signal was set to one of 487 randomly selected bits from the feature vector and the discount rate was again set to one of four timescales. The discount rate $\gamma^{(i)}$ was one of the four values in $\{0, 0.8, 0.95, 0.9875\}$, corresponding to time scales of approximately 0.1, 0.5, 2, and 8 seconds respectively. For each question, the step-size parameter was set to $\alpha = \frac{0.1}{457}$ ($\frac{1}{10}$th of the number of active features), and the trace parameter was set to $\lambda = 0.9$. The initial weight vector was initialized to 0.

An initial performance question was scalability, in particular whether so many predictions can be made and learned in real time. We found that the total computation time for a cycle under our conditions was 55ms, well within the 100ms duty cycle of the robot. The wall-following policy, tile-coding, and the TD(λ) learning algorithm were all implemented in Java and run on a laptop connected to the robot by a dedicated wireless link. The laptop used an Intel Core 2 Duo processor with a 2.4GHz clock cycle, 3MB of shared L3 cache, and 4GB DDR3 RAM. The system garbage collector was called on every timestep to reduce variability. Four threads were used for the learning code. For offline analysis, data was also logged to disk for 120000 timesteps (3 hours and 20 minutes). The total memory consumption was 400MB. Note that with faster computers, the number of predictions or the size of the weight and feature vectors can be increased at least proportionally. This strategy for prediction should scale to millions of predictions with the foreseeable increases in parallel computing power over the next decade.

In the text that follows, we consider a single nexting prediction in detail. Each nexting prediction asks "What will happen next?" over a relatively short, but temporally extended, time scale. Consider the robot's ability to anticipate when one of its light sensors will be saturated as it passes the lamp in one corner of the pen. Examples of returns for different time scales are shown in Fig. 3 (left). The returns for each question are computed from the stored log of observations using Equation 1. For each point in time t, the value of the return constitutes the empirical ground truth answer for the

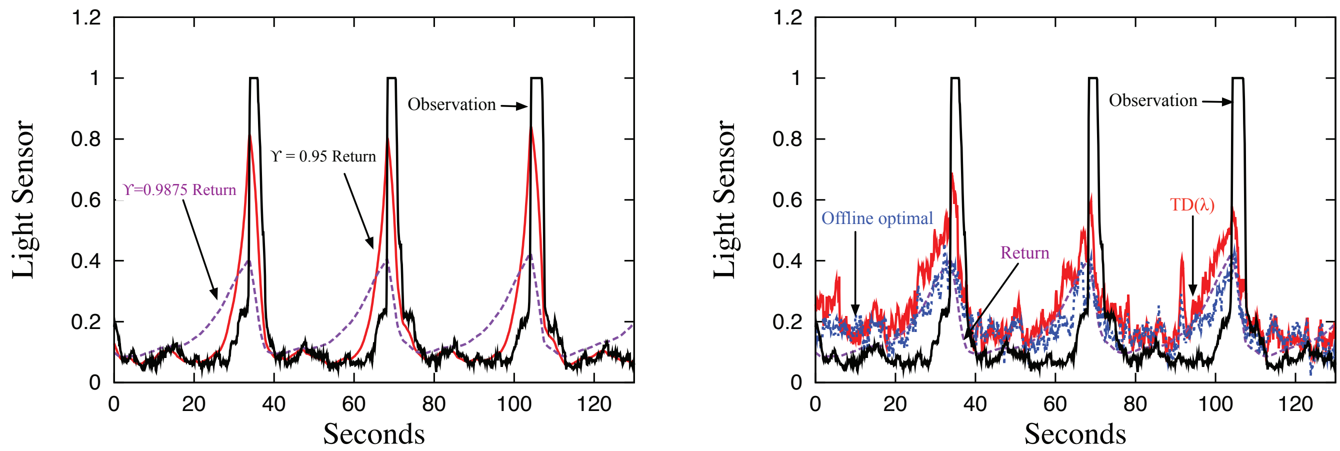

Fig. 3. *(left)* A plot of the returns for one light sensor. The sensor readings exhibit sharp changes when the robot passes the lamp. The returns for each question are computed at the end of the experiment. *(right)* The learned prediction closely matches the return, and the predictions also have the desired qualitative structure of rising in advance of changes in the light signal.

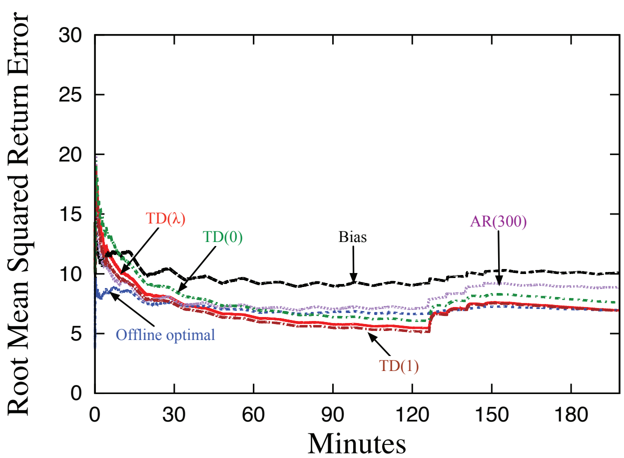

Fig. 4. A comparison of the return errors from different methods for the light prediction question above (8 seconds=80 timesteps). By the end of the dataset, the error of TD(λ) is nearly identical to the error of the best offline weights, and to TD(1). The error of TD(0) is slightly higher, but still less than the error from a bias unit representation, and less than the error of an autoregressive model with a 300-step history. The bump in the error curves (after 120 minutes) coincides with when the robot stalled near the lamp.

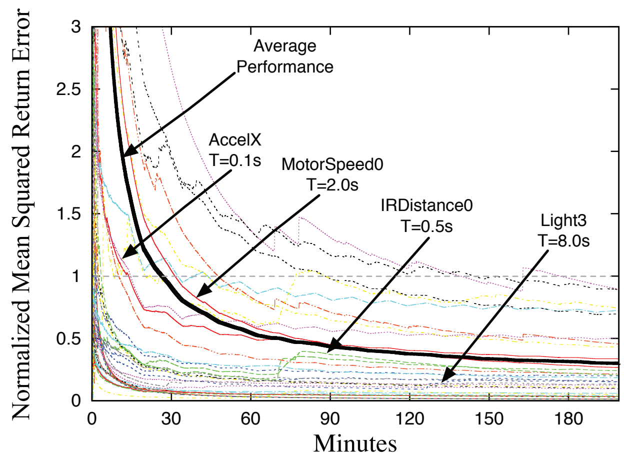

Fig. 5. The presented method learned to answer 2160 questions in real time about future sensor observations generated by the mobile robot's behaviour. The questions pertain to the expected values over the near future at timescales of 0.1, 0.5, 2, and 8 seconds. Only a handful of the learned predictions about sensors and features are shown above; sensors are diverse and include motor temperatures, currents, voltages, light sensors, infrared light sensors, ambient temperature, magnetometers, accelerometers, and others. The predictions are made in real time at a rate of 10Hz. The answers have substantial accuracy; the error shown for each prediction is normalized by the observed variance (Equation 5). Moreover, substantial learning is achieved within the first 30 minutes, as is seen in the error curve for the average performance.

question.

The returns computed offline are compared to the predictions that were made in real time during the experiment in Fig. 3 (right). The predictions in the graph show a clear example of anticipating the increase in light. The return and the learned prediction are in close correspondence. The performance of the learned predictions is also similar to the performance of the best fixed weights θ_*, that was computed offline for the given set of features.

A comparison of the return errors of several methods for this light prediction question is shown in Fig. 4. The graph shows that the TD(λ) algorithm performed well. By the end of the dataset, TD(λ) matched the error of the best offline weights and TD(1). With the same representation as TD(λ), TD(0) had a higher error. Using only a bias unit as a representation (a single active feature) had the highest error of all the methods. We compared these methods to an autoregressive model trained incrementally by the least mean square rule (delaying the learning by 600 timesteps to compute the return). An offline sweep over the autoregressive model parameters gave the lowest error with an autoregressive model of order 300 (which learns a linear predictor with the last 300 light observations as the representation). This model had a higher error than all the TD methods, but less error than a bias unit.

Fig. 5 demonstrates a key result, namely that many accurate answers to predictive questions can be learned in parallel from standard robot behaviour. To compare the accuracy of the different questions, the prediction errors are normalized by the sample variance of the returns for each question over the entire dataset. This yields a normalized mean squared return error (NMSRE),

$$\mathrm{NMSRE}(\hat{g}^i, t) = \frac{1}{t}\frac{\sum_{k=0}^{t}(\hat{g}^i_k - g^i_k)^2}{\mathrm{Var}(g^i)}. \quad (5)$$

The NMSRE value represents the fraction of the variance in the return that remains unexplained by the predictor (the NMSRE is above one initially as the predictors have no data and perform worse than the best constant predictor).

For every question, we can observe in Fig. 5 that the error decreases along an exponential curve. Substantial learning occurs in the first 30 minutes, but errors continue to decrease with additional experience. Note that the error expresses the fraction of the sample variance unexplained and that for every question this falls below 1. Thus, the answers are non-trivial even for a noisy sensor such as an accelerometer at long time scales—the answers learned by the system with the given experience and choice of representation outperform the best constant prediction for every single question.

These results demonstrate our novel and somewhat surprising claim that it is practical to acquire a broad range of knowledge in real time directly from experience. We have shown a method with a sound theoretical foundation that learns answers to thousands of different empirical questions in practice, from regular robot experience. The method is scalable in the number of questions and features, as the amount of computation is linear in each. It is robust in practice, as no individual tuning is required for the different questions. It supports parallel implementation, which was used in dividing the computation across multiple threads. This method provides access to knowledge about multiple timescales while operating at a single fast timescale. This is an impressive set of properties.

Viewed from another perspective, our approach is one way to enable robots to exploit a key insight of machine learning from big data: large, simple, discriminative models will outperform small, complex, generative models when given sufficient data. Substantially more structure often exists in the robot's experience than is predicted by small generative models. This has been seen in several domains, including games, search engines, recommender systems, and even in Jeopardy. The method described here is one simple way to provide a robot with immediate access to knowledge about many facets of its observable existence. This is different from the way knowledge is typically considered on a robot, and opens the door to methods that can leverage a diverse body of knowledge.

IV. An Illustrative Application

Our proposed method generalizes to any robot. To illustrate this point, we describe an application to a robot arm (as described originally by Pilarski et al. (2012)). The domain of human-machine interaction provides a rich source of problems for which predictive empirical knowledge can be interesting,

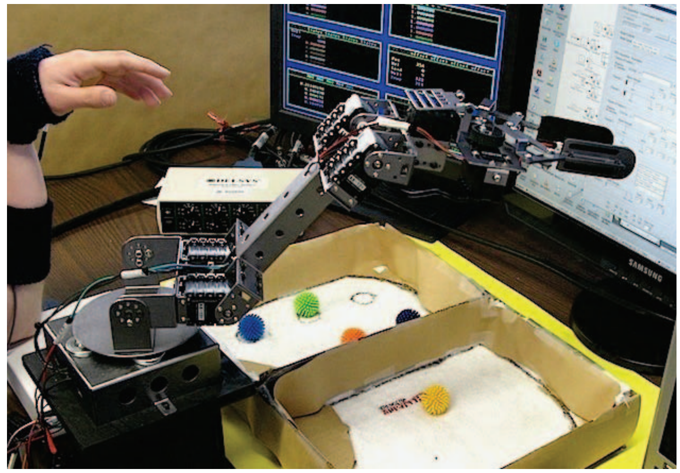

Fig. 6. The myoelectric training tool (MTT), a multi-joint robotic prosthesis used to train new amputees.

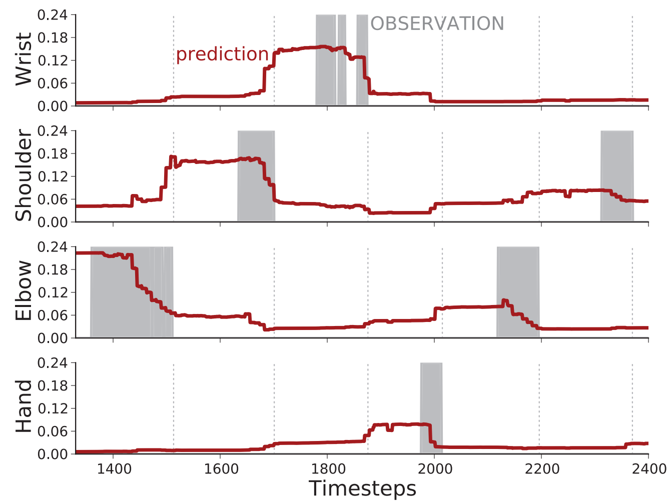

Fig. 7. The user must perform a task, manually switching between four degrees of freedom. The shaded blocks indicate when a joint is active (the observed signal), and solid red lines indicate the system's predictions after less than 15min of online learning; intervals with no activity on any joint are switching times. The grey vertical dotted lines indicate the end of a joint activity and thus the start of a switching period. The system learns to anticipate which degree of freedom the user will move next. These choices will vary across users and even within a task for a single user.

potentially useful, and difficult to acquire by other means. Researchers have developed a robotic platform for familiarizing new amputee patients with the process of controlling a powered prosthetic arm (Dawson et al., 2012). This *myoelectric training tool* (MTT) includes a table-top robotic arm (Fig. 6), which an amputee must learn to control using signals from their remaining muscles.

The MTT enables four degrees of freedom, but the typical lack of distinct recording sites on an amputee patient restricts the number of channels available for motor control. As per standard commercial prostheses, control is therefore multiplexed, with one channel being used to switch between joints in a cyclic order, and a pair of control channels actuating the currently active joint. However, a user can spend an unacceptably large portion of their time selecting which joint they wish to move next with switching-based approaches of this nature, as shown by the time periods with no joint activity in Fig. 7.

We applied nexting predictions to examine if online prediction learning can support user switching in this setting. In these experiments, four predictions were defined, one for the user-driven motion of each joint—the learning agent's goal was to predict which joint the human user would use next. These predictive questions represent temporally extended expectations about motor activity on each of the MTT's four joints. For this task, the robot was operating with a duty cycle of 20ms, and nexting questions were set to have a timescale of 2.5 seconds ($\gamma = 0.992$). The feature vector used by the learning system was generated by jointly tiling all four robot joint angles with each of the other 28 sensors provided by the MTT system. The resulting feature vector was sparse, consisting of 1,306,369 features, of which 169 were active at any given time.

Fig. 7 shows the results of the nexting predictions after 15 minutes of online learning. The predictions consistently anticipate which joint the user will move next, as shown by the result that the joint that is selected next often has the greatest magnitude of the four predictions. This information could be used to change the joint selection ordering, so that instead of cycling through a fixed order, the system would cycle through the joints in an order given by the magnitude of the nexting predictions. Nexting-based joint selection of this kind was found to decrease the number of switching commands that a user would have to provide, and thus the total projected time used for transitions, as calculated using the mean times observed for transitions involving one, two, or three user switching actions (Fig. 8). The projected transition cost for the adaptive order was then compared to the cost for the best possible fixed switching order, as computed post-hoc from the recorded data. Based on this comparison, it was found that nexting predictions could facilitate a switching time decrease of more than 14% on this task (Fig. 8). Moreover, the time taken for switching was found to rise monotonically with the number of switch commands. This means that with adequate feedback to the user, this predictive approach could reduce the amount of real time a patient spends on tasks.

This approach to adapting the switching order demonstrates one direct benefit of learning *in situ*. This is a scenario where, even though a person is always in control of the actions being performed, the robot can make the user's life better by learning to anticipate what the user will want next. As shown in these results, the best fixed ordering for the task is outperformed by an adaptive ordering. Given the fact that a user will switch between several tasks and can solve the same task in different ways, it is difficult to see how a non-adaptive approach could achieve the same benefits. Moreover, in spite of the relatively large number of variables and the large feature space, learning is still computationally and data efficient, as this level of performance is reached in 15 minutes. All learning related computations were completed within 5ms per iteration on a similar laptop to the one used in the earlier experiment.

Transition with 1 switching action, mean time:	1.09 sec
Transition with 2 switching actions, mean time:	1.75 sec
Transition with 3 switching actions, mean time:	2.21 sec
Net experiment time:	20.66 min
Net observed transition time:	10.40 min
Net transition time (projected for best fixed order):	9.98 min
Net transition time (projected for adaptive order):	8.49 min

Potential time savings with adaptive control:	1.49 min
Potential time savings on transitions:	14.3%
Potential time savings on full experiment:	7.2%

Fig. 8. Additional performance numbers for the switching task, including projected time savings from applying nexting predictions.

V. Related Work

Much previous work on reinforcement learning for real-time robotics has focused on its role in control. For example, the Natural Actor-Critic algorithm (Peters and Schaal, 2008) has a critic that makes a single prediction. Other reinforcement learning approaches focus on policy evaluation without a predictive component, as used for example in improving a quadruped walk (Kohl and Stone, 2004). Previous work with reinforcement learning on robots has not demonstrated learning of thousands of temporally extended predictions in real time.

Related to the idea of learning many predictions in parallel is the idea of constructing optimal predictions for a set of tests (Talvitie and Singh, 2011). The domains differ greatly however, as in that work the emphasis is on constructing the most accurate predictive answers for partially observable systems that have a small discrete set of observations, whereas this paper is concerned with satisfying the constraints of learning in real time with continuous data on a robot.

The most similar work to the current system is an online variant of an offline spectral method for making many temporally extended predictions (Boots et al., 2011). Their work differs from the work presented here in several important ways. Although their algorithm is incremental and online, supporting real-time operation is non-trivial because their algorithm uses computationally expensive matrix operations and sophisticated data transformations. Their algorithm requires a window of past and future observations, which imposes additional memory requirements. Finally, their algorithm strongly couples the various prediction questions to discover joint structure, and coupling the problems together in this way prevents direct parallel implementations. Despite these important distinctions in implementation between the two methods, it is possible that ideas from both methods can be fruitfully combined because of their similar learning objectives.

A standard approach for predicting the temporally extended consequences of behaviour is to use model-based roll-outs of a one timestep dynamical model. This requires the acquisition of a one timestep model of robot dynamics, often analytically or offline from logs (Thrun and Mitchell, 1995), and then using this model for adapting control. Another line of work uses online, real-time learning with a large memory of past experiences to dynamically construct local models (Atkeson et al., 1997). However, that work does not develop the use of local models for temporally extended predictions about many different targets. Moreover, their approach is computationally intractable for systems that lack an underlying low dimensional description, such as a mobile robot with a diverse set of sensing modalities.

Previous work has relied on small one timestep models to manage the computational expense of using roll-outs to generate temporally extended predictions. One approach used roll-outs of a one timestep model to ensure that a robot with substantial inertia can both move quickly and stop safely (Fox et al., 1997). Recent work on real-time control of an autonomous helicopter (Abbeel et al., 2010) uses both a cost-to-go function (a variant of a value function) and simulations of the system dynamics at 20Hz with a two-second horizon.

A major distinction from our work is that this standard approach needs to be given an accurate model of the dynamics. The dynamics of a robot's interaction with the environment can be poorly understood prior to deployment, particularly when robots are sent into novel domains including underwater exploration, space, disasters, and human bodies. Our approach enables a robot to learn partial models of the relevant dynamics directly from its stream of real-time interaction. A robot with the ability to acquire knowledge of its dynamics from experience can benefit both designers and end-users of the robot through additional flexibility and improved performance.

VI. Broadening the Space of Questions (Ongoing and Future Work)

We have demonstrated a method that learns to make thousands of temporally extended predictions directly from robot experience in real time. The predictions are answering questions about the future that are empirical and multi-scale, but are of a more constrained form than can be answered by simulation roll-outs with a one-timestep model. In this section, we outline generalizations of our approach that substantially broaden the space of predictions that can be expressed. The generalizations are based on the theory of options (Sutton et al., 1999) and on allowing general value functions to be option conditional (Sutton et al., 2011).

The first generalization is to permit γ to vary with the state, $\gamma = \gamma_t$, which enables questions with state based (pseudo)-termination to be posed. For example the question "How many timesteps will elapse until all the wheels stop turning?" can be posed by setting $r_t = 1$ and $\gamma_t = 0$ if the observed velocity of every wheel is zero and $\gamma_t = 1$ otherwise. As another example, Fig. 9 shows an example of the amount of power consumed until a light sensor is saturated or approximately two seconds have elapsed, where $\gamma_t = .95 \times \mathbb{I}_{\text{Light3<Saturation}}$, and

$$r_t = \Sigma_{i=1}^{3} \text{MotorCurrent}i_t \times \text{MotorVoltage}i_t.$$

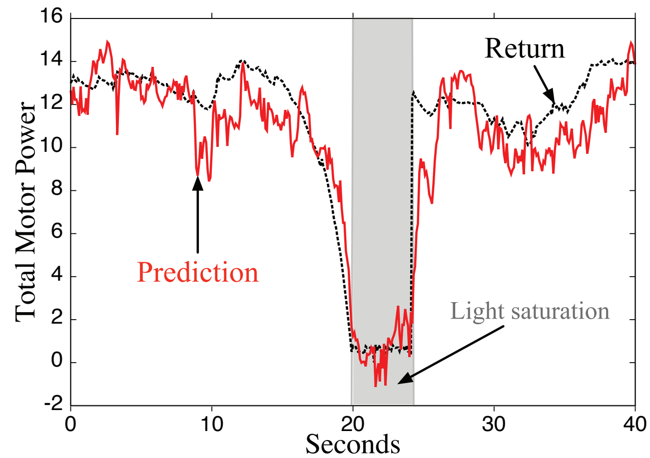

Fig. 9. TD(λ) can learn to answer questions where γ varies with time for state based terminations and the addition of terminal outcomes. Here, we see predictions matching returns for the question of how much power will be used until the sensor Light3 is saturated, or spontaneous termination occurs with a 2 second horizon.

This prediction was learned with the same parameters and feature representations used in our result in Section III.

The next generalization is to add to the return an outcome z_t at termination. This allows questions to be expressed where the final state is relevant. For example, the robot's expected temperature on Motor2 when the Light3 sensor is saturated can be expressed by setting $r_t = 0$, $\gamma_t = .95 \times \mathbb{I}_{\text{Light3<Saturation}}$, $z_t = \text{MotorTemperature2}$. Incorporating z_t into the return is supported by standard TD(λ). For all the above generalizations of questions, the error in the learned predictions should decrease at rates comparable to those shown earlier.

A final generalization is to consider questions about different ways of behaving. If the robot behaves according to one policy, it is challenging to learn about the consequences of following a different policy. In this *off-policy* learning setting, the standard TD(λ) algorithm can diverge. The expression for the general value function g is modified,

$$g^{r,z,\gamma,\pi}(s) = E[G_t^{r,z,\gamma,\pi} | S_t = s, S_{t+1} \sim \pi(S_t)], \tag{6}$$

and the return G is extended,

$$G_t^{r,z,\gamma,\pi} = r_{t+1} + \ldots + r_T + z_T, \tag{7}$$

where the termination time T is sampled according to γ. This general question form is known as option-conditional prediction (Sutton et al., 1999).

As an example of an off-policy scenario, consider the sensory experience of a car being driven by a person being used to learn to predict the time to come to a complete stop while braking. Many environmental aspects could influence stopping times while braking, including gravel roads, temperature, and rain. The ability to learn off-policy enables learning from all the snippets of experience when the driver touches the brakes, and not just the times when the car comes to a complete stop.

To learn to answer off-policy questions in real time, one can use the GTD(λ) algorithm (Maei, 2011) that uses the ratio

$\rho_t = \frac{\pi(A_t|S_t)}{\pi_b(A_t|S_t)}$ in the update equations that are shown below.

$$\delta_t = r_{t+1} + (1-\gamma_{t+1})z_{t+1} + \gamma_{t+1}\theta_t^\top x_{t+1} - \theta_t^\top x_t \quad (8)$$

$$e_t = \rho_t(x_t + \gamma_t \lambda_t e_{t-1}) \quad (9)$$

$$\theta_{t+1} = \theta_t + \alpha(\delta_t e_t - \gamma_{t+1}(1-\lambda_{t+1})(e_t^\top w_t)x_{t+1}) \quad (10)$$

$$w_{t+1} = w_t + \beta(\delta_t e_t - (x_t^\top w_t)x_t) \quad (11)$$

The primary computational differences between GTD(λ) and TD(λ) are an additional weight vector w, an associated step-size parameter β, and an explicit computation of the ratio ρ between π_b (the robot's behaviour policy) and π (the policy considered by the prediction).

This algorithm is a gradient-based generalization of the traditional TD(λ) algorithm. The algorithm learns an answer to the question g specified by r,z,π, and γ as a linear function of the feature vector, with the same form of linear prediction, $\hat{g}(x_t) = \theta_t^\top x_t$, and thus the same linear complexity. The GTD(λ) algorithm maintains guarantees of stability when learning off-policy and converges to a fixed-point that minimizes the mean-squared projected Bellman error weighted by the distribution of states visited by the behaviour (Maei, 2011). This somewhat technical objective is a natural one for online learning, as the algorithm minimizes the error that arises from the agent's limited perception (projecting environmental state onto the feature vector x), for the Bellman error (the difference between prediction and reality across adjacent timesteps), for the experience generated by the robot's behaviour that is relevant to the policy considered by the prediction.

The space of questions that can be expressed in this final setting is quite general, and possibly covers all the interesting temporally extended predictions that one can answer with one-timestep models (Sutton et al., 2011). However, there remain numerous complications introduced by the general setting that make it unsuitable for the demonstration of scaling that is the focus of the present work. In particular, it is difficult to directly measure performance in an off-policy setting. In an off-policy setting, predictions are made about many different ways of behaving, but from each state only the predictions for one way of behaving can be tested at a time. Moreover, performing a test alters the state and the state distribution from which experience is gathered. These issues can likely be addressed, and we plan to explore this direction in future work.

VII. Conclusions

We have demonstrated that a robot can learn to answer temporally extended predictive questions in real time at scale—for thousands of questions, using thousands of features, with amounts of experience and computation that are commonly available on robots today. This approach provides a principled technique for a robot to acquire knowledge from experience in real time about the temporally extended consequences of its behaviour. We have described one potential use for this style of knowledge as part of an adaptive user interface for a robot arm. The method is straightforward to deploy on different robots, and presents promising directions for future study.

Acknowledgments

The authors thank Mike Sokolsky for creating the Critterbot, and Thomas Degris for essential assistance with the experiments briefly reported in Sections 4 and 6. The authors also thank Michael R. Dawson and Jason P. Carey for their contributions to the myoelectric control study in Section 4.

References

Abbeel, P., Coates, A., and Ng, A. Y. (2010). Autonomous helicopter aerobatics through apprenticeship learning. *International Journal of Robotics Research (IJRR)*.

Atkeson, C. G., Moore, A. W., and Schaal, S. (1997). Locally weighted learning. *Artificial Intelligence Review*, 11(1/5):11–73.

Boots, B., Siddiqi, S., and Gordon, G. (2011). An online spectral learning algorithm for partially observable nonlinear dynamical systems. In *Proceedings of the Twenty-Fifth National Conference on Artificial Intelligence (AAAI)*.

Dawson, M. R., Fahimi, F., and Carey, J. P. (2012). The development of a myoelectric training tool for above-elbow amputees. *The Open Biomedical Engineering Journal*, in press.

Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1).

Kohl, N. and Stone, P. (2004). Machine learning for fast quadrupedal locomotion. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI)*, pages 611–616.

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press.

Maei, H. R. (2011). *Gradient Temporal-Difference Learning Algorithms*. PhD thesis, University of Alberta.

Modayil, J., White, A., and Sutton, R. S. (2012). Multi-timescale nexting in a reinforcement learning robot. In *Proc. of the Int. Conf. on Simulation of Adaptive Behaviour (to appear)*.

Peters, J. and Schaal, S. (2008). Natural actor-critic. *Neurocomputing*, 71:1180–1190.

Pilarski, P. M., Dawson, M. R., Degris, T., Carey, J. P., and Sutton, R. S. (2012). Dynamic switching and real-time machine learning for improved human control of assistive biomedical robots. In *Proceedings of the 4th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 296–302.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44.

Sutton, R. S., Precup, D., and Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., and Precup, D. (2011). Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Talvitie, E. and Singh, S. (2011). Learning to make predictions in partially observable environments without a generative model. *Journal of Artificial Intelligence Research*.

Thrun, S. and Mitchell, T. (1995). Lifelong robot learning. *Robotics and Autonomous Systems*.