Multi-timescale Nexting in a Reinforcement Learning Robot

Joseph Modayil, Adam White, and Richard S. Sutton

Reinforcement Learning and Artificial Intelligence Laboratory
Department of Computing Science, University of Alberta

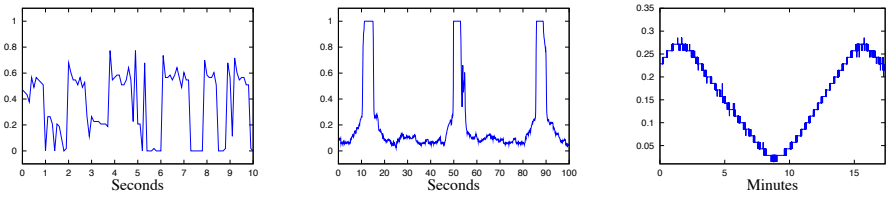
December 14, 2011

Abstract

The term "nexting" has been used by psychologists to refer to the propensity of people and many other animals to continually predict what will happen next in an immediate, local, and personal sense. The ability to "next" constitutes a basic kind of awareness and knowledge of one's environment. In this paper we present results with a robot that learns to next in real time, predicting thousands of features of the world's state, including all sensory inputs, at timescales from 0.1 to 8 seconds. This was achieved by treating each state feature as a reward and applying temporal-difference methods to learn a corresponding value function with a discount rate corresponding to the timescale. That is, instead of predicting a single distinguished reward on a long timescale, as in conventional reinforcement learning, we predicted many state features at multiple short timescales. Although this approach is conceptually straightforward, there are many computational and performance challenges in implementing it in real time on a physical robot. We show that two thousand predictions, each dependent on six thousand state features, can be learned and updated online at better than 10Hz on a laptop computer, using the standard TD(λ) algorithm with linear function approximation. We show that this approach is efficient enough to be practical, with much of the learning complete within 30 minutes. We also show that a single tile-coded feature representation suffices to accurately predict many different signals at a significant range of timescales. Finally, we show that the accuracy of our learned predictions compares favorably with the optimal off-line solution, and with conventional auto-regressive prediction methods.

1 Multi-timescale Nexting

Many psychologists have noted that people and other animals seem to continually make large numbers of short-term predictions about their sensory input (e.g., see Gilbert 2006, Brogden 1939, Pezzulo 2008, Carlsson et al. 2000).



(a) Motor Current (b) Light (c) Motor Temperature

Figure 1: Examples of robot sensory signals varying over different time scales: (a) motor current varying over tenths of a second, (b) an ambient light sensor varying over seconds, and (c) a motor temperature sensor varying over tens of minutes.

When we hear a melody we predict what the next note will be or when the next downbeat will occur, and are surprised and interested (or annoyed) when our predictions are disconfirmed (Huron 2006, Levitin 2006). When we see a bird in flight, hear our own footsteps, or handle an object, we continually make and confirm multiple predictions about our sensory input. When we ride a bike, ski, or rollerblade, we have finely tuned moment-by-moment predictions of whether we will fall, and of how our trajectory will change in a turn. In all these examples, we continually predict what will happen to us *next*. Making predictions of this simple, personal, short-term kind has been called *nexting* (Gilbert, 2006).

Nexting predictions are specific to one individual and to their personal, immediate sensory signals or state variables. A special name for these predictions seems appropriate because they are unlike predictions of the stock market, of political events, or of fashion trends. Predictions of such public events seem to involve more cognition and deliberation, and are fewer in number. In nexting we envision that one individual may be continually making massive numbers of small predictions in parallel. Moreover, nexting predictions seem to be made simultaneously at multiple time scales. When we read, for example, it seems likely that we next at the letter, word, and sentence levels, each involving substantially different time scales. Figure 1 shows examples of three sensory signals from our robot that have predictable regularities at vastly different time scales.

Many scientists have proposed that the ability to predict and anticipate is a key part of intelligence (e.g., Tolman 1951, Hawkins & Blakeslee, 2004, Butz et al. 2003, Wolpert et al. 1995). Nexting can be seen as the most basic kind of prediction, preceding and possibly underlying all the others. That people and a wide variety of animals learn and make simple predictions at a range of short time scales in conditioning experiments was established so long ago that it is known as *classical conditioning* (Pavlov 1927). Predictions of upcoming shock to a paw may reveal themselves in limb-retraction attempts a fraction of a second before the shock, and as increases in heart rate 30 seconds prior. In other experiments, for example those known as *sensory preconditioning* (Brogden 1939, Rescorla 1980), it has been clearly shown that animals learn predictive relationships between stimuli even when none of them are inherently good or

bad (like food and shock) or connected to an innate response. In this case the predictions are made, but not expressed in behavior until some later experimental manipulation connects them to a response. Animals seem to just be wired to learn the many predictive relationships in their world.

To be able to next is to have a basic kind of knowledge about how the world works in interaction with one's body. It is to have a limited form of forward model of the world's dynamics. To be able to learn to next—to notice any disconfirmed predictions and continually adjust your nexting—is to be aware of one's world in a significant way. Thus, to build a robot that can do both of these things is a natural goal for artificial intelligence. Prior attempts to achieve artificial nexting can be grouped in two approaches. The first approach is to build a *myopic* forward model of the world's dynamics, either in terms of differential equations or state-transition probabilities (e.g., Wolpert et al. 1995, Grush 2004, Sutton and Barto 1990). In this approach a small number of carefully chosen predictions are made of selected state variables with a public meaning. The model is myopic in that the predictions are only short term, either infinitesimally short in the case of differential equations, or maximally short in the case of the one-step predictions of Markov models. In these ways, this approach has ended up in practice being very different from nexting. The second approach, which we follow here, is to use temporal-difference (TD) methods to learn long-term predictions directly. The prior work pursuing this approach has almost all been in simulation, and has used table-lookup representations and a small number of predictions (e.g., Sutton 1995, Kaelbling 1993, Singh 1992, Sutton et al. 1999, Dayan and Hinton 1993). Sutton et al. (2011) showed real-time learning of TD predictions on a robot, but did not demonstrate the ability to learn many predictions in real time or with a single feature representation.

2 Nexting as Multiple Value Functions

We take a reinforcement-learning approach to achieving nexting. In reinforcement learning it is commonplace to learn long-term predictions of reward, called *value functions*, and to learn these using temporal-difference (TD) methods such as TD(λ). However, TD(λ) has also been used as a model of classical conditioning, where the predictions are shorter term and where more than one signal might be viewed as a reward. Our approach to nexting can be seen as taking this latter approach to the extreme of predicting massive numbers of signals of all kinds at multiple time scales.

We use a notation for our multiple predictions that mirrors—or rather multiplies—that used for conventional value functions. Time is taken to be discrete, $t = 1, 2, 3, \ldots$, with each time step corresponding to approximately 0.1 seconds of real time. Our ith prediction at time t, denoted v_t^i, is meant to anticipate the future values of the ith prediction's designated "reward" signal, r_t^i, over a designated time scale given by the discount-rate parameter γ^i. In our experiments, the target signal r_t^i was either a raw sensory signal or else a component of a state-feature vector (that we will introduce shortly), and the

discount-rate parameter was one of four fixed values. The goal of learning is for each prediction to approximately equal the correspondingly discounted sum of the future values of the corresponding reward signal:

$$v_t^i \approx \sum_{k=0}^{\infty} (\gamma^i)^k r_{t+k+1}^i \stackrel{\text{def}}{=} G_t^i. \tag{1}$$

The random quantity G_t^i is known as the *return*.

We use linear function approximation to form each prediction. That is, we assume that the state of the world is characterized at each time by a feature vector ϕ_t, and that all the predictions v_t^i are formed as inner products of ϕ_t with the corresponding weight vectors $\theta_t^i \in \mathbb{R}^n$:

$$v_t^i = \phi_t^\top \theta_t^i \stackrel{\text{def}}{=} \sum_j \phi_t(j)\theta_t^i(j), \tag{2}$$

where $\phi_t^\top$ denotes the transpose of ϕ_t (all vectors are column vectors unless transposed) and $\phi_t(j)$ denotes its jth component. The predictions at each time are thus determined by the weight vectors θ_t^i. One natural algorithm for learning the weight vectors is linear TD(λ):

$$\theta_{t+1}^i = \theta_t^i + \alpha \left(r_{t+1}^i + \gamma^i \phi_{t+1}^\top \theta_t^i - \phi_t^\top \theta_t^i \right) \mathbf{e}_t^i \tag{3}$$

where $\alpha > 0$ is a step-size parameter and $\mathbf{e}_t^i \in \mathbb{R}^n$ is an *eligibility trace* vector, initially set to zero and then updated on each step by

$$\mathbf{e}_t^i = \gamma^i \lambda \mathbf{e}_{t-1}^i + \phi_t, \tag{4}$$

where $\lambda \in [0, 1]$ is a trace-decay parameter.

Under common assumptions and a decreasing step-size parameter, TD(λ) with $\lambda = 1$ converges asymptotically to the weight vector that minimizes the mean squared error between the prediction and its return. In practice, smaller values of $\lambda \in [0, 1)$ are almost always used because they can result in significantly faster learning (e.g., see Sutton & Barto 1998), but the $\lambda = 1$ case still provides an important theoretical touchstone. In this case we can define an optimal weight value θ_*^i that minimizes the squared error from the return over the first N predictions:

$$\theta_*^i = \arg\min_\theta \sum_{t=1}^{N} \left(\phi_t^\top \theta - G_t^i \right)^2$$

This value can be computed offline by standard algorithms for solving large least-squares regression problems, and the performance of this offline-optimal value can be compared with that of the weight vectors found online by TD(λ). The offline algorithm is $O(n^3)$ in computation and $O(n^2)$ in memory, and thus is just barely tractable for the cases we consider here, in which $n = 6065$. Nevertheless, θ_*^i provides an important performance standard in that it provides an upper limit on one measure of the quality of the predictions found by learning. This

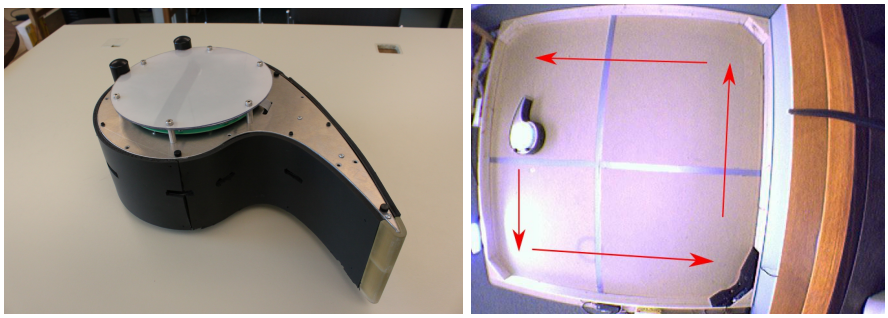


Figure 2: Our platform for these experiments is a mobile robot (left) with multiple sensors that gathers experience while wall-following in its pen (right). This experience contains observations of both stochastic events (such as ambient light variations from the sun and shadows) and regular events (such as passing a lamp on the lower-left side of the pen).

upper limit is determined not by any learning algorithm, but by the feature representation. As we will see, even the predictions due to θ^i_* will have residual error. Thus, this analysis provides a method for determining when performance can be improved with more experience and when performance improvements require a better representation. Note that this technique is applicable even when experience is gathered from the physical world, where no formal notion of state is available.

3 Experimental Setup

We investigated the practicality of nexting on our sensor-rich mobile robot platform (Figure 2). Our custom-designed mobile robot has a diverse set of sensors and has holonomic motion provided by three omni-wheels. Both the sensor and motor subsystems provide low-level software access. Action commands are transformed into voltage signals by a motor subsystem with on-board current-limiting and overheating protection. Sensors attached to the motors report the electrical current, the input motor voltage, motor temperature, wheel rotational velocities, and an overheating flag, providing substantial observability of the internal physical state of the robot. In addition to the sensors that measure the internal system, other sensors collect information from the external environment. Passive sensors detect ambient light in several directions from the top of the robot in the visible and infrared spectrum. Active sensors emit infrared light and measure the reflectance on the sides of the robot (which provides information about the distance to nearby obstacles). Other sensors report acceleration, rotation, and the magnetic field. In total, we consider 53 different sensor readings, all normalized to values between 0 and 1 based on sensor limits.

For our experiments, the agent's representation was constructed via tile coding. This produced a binary vector, $\phi_t \in \{0,1\}^n$, with a constant number of

1 features (see Sutton & Barto 1998). The features provided no history and performed no averaging of sensor values. The tile coder was comprised of many overlapping tilings of single and pairs of sensors (see Table 1). 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.

The robot experiment was conducted in a square wooden pen, approximately two meters on a side, with a lamp on one edge. The robot's actions were selected according to a fixed stochastic wall-following policy. This policy moved forward by default, slid left or right to keep a side IRDistance sensor within a bounded range (50-200), and drove backward while turning when the front IRDistance sensor reported a nearby obstacle. The robot completed a loop of the pen approximately once every 40 seconds. Due to overheating protection, the motors would stop to cool down at approximately 14 minute intervals. To increase the diversity of the data, the policy selected an action at random with a probability $p = 0.05$. At every time step (approximately 100ms), sensory data was gathered and an action performed. This simple policy was sufficient for the robot to reliably follow the wall for hours, even with overheating interruptions.

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 time step to reduce variability. Four threads were used for the learning code. For offline analysis, data was also logged to disk for 120000 time steps (3 hours and 20 minutes).

4 Results

We applied TD(λ) to learn 2160 predictions. For the first 212 predictions, the reward signal, r_t^i, was the sensor reading of one of the 53 sensors listed in Table 1, and the discount rate, γ^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 re-

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)

Table 1: 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 done for each sensor or sensor group, and how many intervals (resolution) were involved in each dimension of each tiling. See text for explanation.

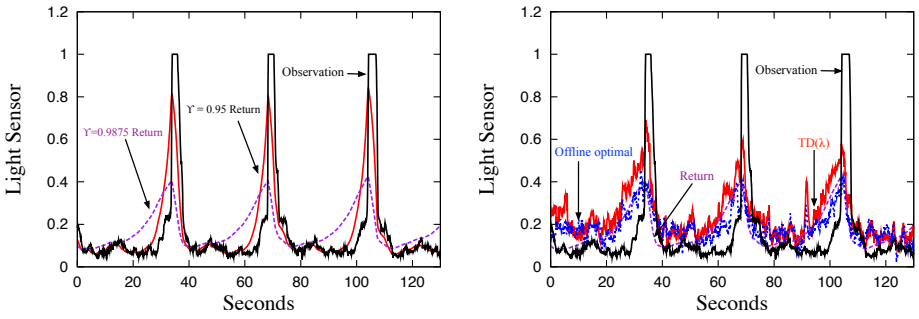


Figure 3: Comparison of ideal (left) and learned (right) predictions of one of the light sensors for three trips around the pen after 2.5 hours of experience. On each trip, the raw sensor value saturates at 1.0. The returns for the 2 and 8-second predictions, shown on the left, rise in anticipation of the high value, and then fall in anticipation of the low value. The 8-second predictions in the second panel of the offline-optimal weights (dotted blue line) and the TD(λ)-learned weights (solid red line) behave similarly both to each other and to the returns (albeit with more noise).

spectively. For the remaining 1948 predictions, the reward was one of the 6065 components of the feature vector ϕ_t, selected at random, and a discount rate of one of the four values in $\{0, 0.8, 0.95, 0.9875\}$. The learning parameters were $\lambda = 0.9$ and $\alpha = 0.1/457 (= \#$ of active features). The initial weight vector was set to zero.

Our initial performance question was scalability, in particular whether so many predictions could 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 total memory consumption was 400MB. Note that with faster computers the number of predictions or the size of the weight and feature vectors could be increased at least proportionally. This strategy for nexting should be easily scalable to millions of predictions with foreseeable increases in parallel computing power over the next decade.

Of course, rapid running time is no consolation if the learned predictions are inaccurate. To begin assessing accuracy, we took a close look at some of the predictions, in particular, at the prediction for the Light3 sensor at the 8 second timescale. As the robot approaches the lamp in the lower left of the pen on each trip around the pen, this sensor undergoes a regular pattern of increase to saturation and then falling back to a low level, as shown by the labelled lines in Figure 3. If the robot can use its sensors to inform itself where it is in the cycle, then it should be able to anticipate the rising and falling of this sensor value. The first panel of the figure shows the empirical returns at the 8 and 2-second time scales, computed after the fact when the entire future is known. These can be considered the ideal predictions, but of course this ideal is achievable only with knowledge of the future, whereas the predictions must

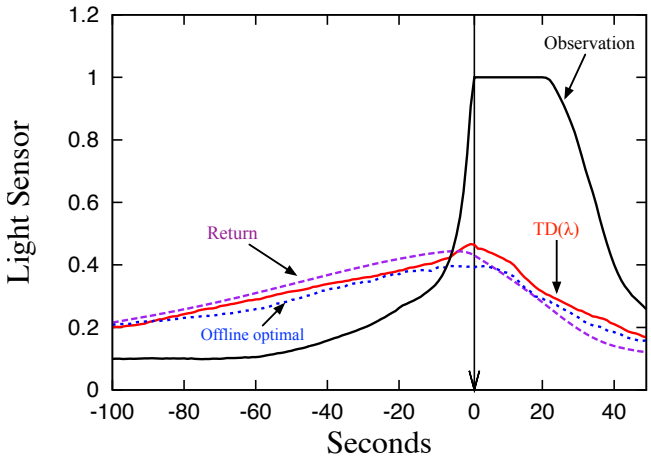


Figure 4: An average of 100 cycles like the three shown in Figure 3 (right panel), aligned on the onset of sensor saturation. Error bars are slightly wider than the lines themselves and overlap substantially, so are dropped for clarity

be made as a (linear) function of the feature vector at the current time.

The second panel of the figure shows the predictions made by TD(λ) at the time and the predictions of the offline-optimal weight vector, θ^i_*, computed afterwards (both for the 8-second time scale). The key result is that the agent has learned to anticipate the onset of increasing light. The learned prediction and the optimal prediction match the empirical return closely, though with substantial noisy perturbations.

Figure 4 is a still closer look at this same prediction, obtained by averaging over 100 circuits around the pen, aligning each circuit's data so that the time of initial saturation of the light sensor is the same. We can now see very clearly how the predictions and returns anticipate both the rise and fall of the sensor value, and that both the TD(λ) prediction and the optimal prediction, when averaged, closely match the return.

Having demonstrated that accurate prediction is possible, we now consider the rate of learning in Figure 5. The graphs shows that learning is fast in terms of data (despite the large number of features), converging to solutions with low error in the familiar exponential way. This result is important as is demonstrates that learning online in real time is possible on robots with a few hours of experience, even with a large distributed representation. For contrast, we also show the learning curve for a trivial representation consisting only of a bias unit (the single feature that is always 1). The comparison serves to highlight that large informative feature sets are beneficial. The comparison to the predictive performance of the offline-optimal solution shows a vanishing performance gap by the end of the experiment. The second panel of the figure shows a similar pattern of decreasing errors for a sample of the 2160 TD(λ) predictions, showing that learning many predictions in parallel yields similar results.

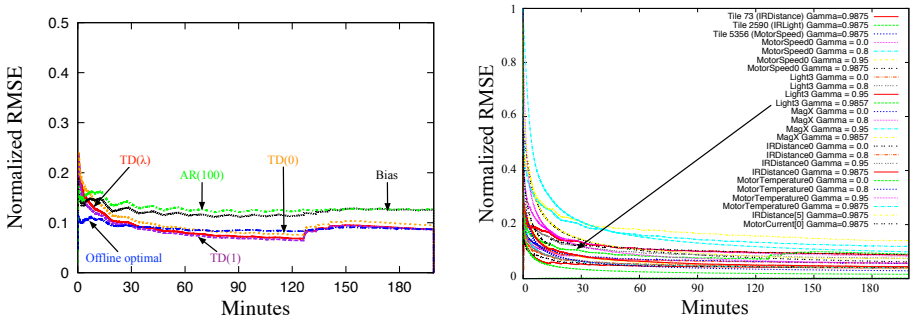


Figure 5: Nexting learning curves for the 8-second light sensor predictions (left) and for a representative sample of the TD(λ) predictions (right). Predictions at different time scales have had their root mean squared error (RMSE) normalized by $\frac{1}{1-\gamma^i}$. The graph on the left is a comparison of different learning algorithms. The jog in the middle of the first graph occurs when the robot stops by the light to cool off its motors, causing the online learners to start making poor predictions. In spite of the unusual event, the TD(λ) solution still approaches the offline-optimal solution. TD(λ) performs similarly to a supervised learner TD(1), and they both slightly outperform TD(0). The curve for the bias unit shows the poor performance of a learner with a trivial representation, and the 100th order autoregressive model (AR(100)) performs quite poorly. The graph on the right shows that seemingly all the TD(λ) predictions are learning well with a single feature representation and a single set of learning parameters.

A noteworthy result is that the same learning parameters and representation suffice for learning answers to a wide variety of nexting predictions without any convergence problems. Although the answers continue to improve over time, the most dramatic gains were achieved after 30 minutes of real time.

5 Discussion

These results provide evidence that online learning of thousands of nexting predictions on a robot in parallel is possible, practical, and accurate. Moreover, the predictive accuracy is reasonable with a few hours of robot experience, even when sharing learning parameters and a large, sparse, feature representation. The parallel scalability by which knowledge is acquired in this process is substantially novel when compared with the predominately sequential existing approaches common for robot learning. These results also show that online methods can be competitive in accuracy with an offline optimization of mean squared error.

The ease with which a simple reinforcement learning algorithm enables nexting on a robot is somewhat surprising. Although the formal theories of reinforcement learning sometimes give mathematical guarantees of convergence,

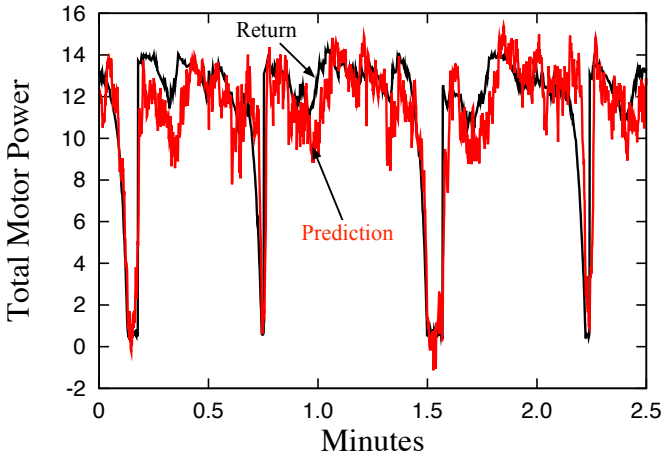


Figure 6: Nexting can be extended, for example to consider time-varying gamma to predict of the amount of power that the robot will expend before a probabilistic pseudo-termination with a 2-second time horizon or a saturation event on Light3.

there is little guidance for the choice of features for a task, for selecting learning parameters across a range of tasks, or for how much experience is required before a reinforcement learning system will approach convergence. The experiments show that we can use the same features across a range of tasks, anticipate events before they occur, and achieve predictive accuracy approaching that of an offline-optimal solution with a limited amount of robot experience.

The exponentially discounted predictions that we have focused on in this paper constitute the simplest kind of nexting. They are a natural first kind of predictive knowledge to be learned. Online TD-style algorithms can be extended to handle a much broader set of predictions, including time-varying choices of γ, time-varying λ, and even off-policy prediction. Perhaps everything that could be learned by observing the stream of experience of a dynamical system might be captured by considering a large enough set of predictions.

As one example of such an extension, consider allowing the discount rate γ^i to vary as a function of the agent's state. The algorithmic modifications required are straightforward. In the definition of the return in Equation 1, $(\gamma^i)^k$ is replaced with $\Pi_{j=0}^{k}\gamma_{t+j}^i$. In Equation 3, γ^i is replaced with γ_{t+1}^i and finally, in Equation 4, γ^i is replaced with γ_t^i. Using the modified definitions, the robot can predict how much motor power it will consume until either Light3 is saturated or approximately two seconds elapse. This prediction can be formalized by setting the prediction's reward signal to be the sum of instantaneous power consumption of each wheel, ($r = \sum_{i=1}^{3} \text{MotorVoltage}i \times \text{MotorCurrent}i$) and throttling gamma when Light3 is saturated ($\gamma_t^i = 0.1$ when Light3 is saturated and 0.95 otherwise). The plots in Figure 6 shows that the robot has learned to anticipate how much power will be expended prior to reach the light or spontaneously terminating.

The knowledge acquired by nexting has the benefit of being adapted to the robot's experience and being fundamentally empirical. The target of the learning algorithms is determined solely by the agent's feature representation, and it is not constrained by assumptions of an underlying generative process. In spite of the lack of a model, this approach still enables offline comparisons with an empirical return and an optimal offline solution. As such it provides practitioners with insight into whether additional experience or additional features would be the most beneficial for improving predictive performance.

6 Conclusions

We have demonstrated multi-timescale nexting on a physical robot; thousands of anticipatory predictions at various time-scales can be learned in parallel on a physical robot in real-time using a reinforcement learning methodology. This approach uses a large feature representation with an online learning algorithm to provide an efficient means for making parallel predictions. The algorithms are capable of making real-time predictions about the future of the robot's sensors at multiple time-scales using the computational horsepower of a laptop. Finally, we have demonstrated the potential value of a single shared representation for solving many tasks in a domain, and a practical role for tuning-free algorithms. Future work will extend these results to more general predictions and control behaviours.

References

Brogden, W. (1939). Sensory pre-conditioning. *Journal of Experimental Psychology*, 25(4):323–332.

Butz, M., Sigaud, O., and Gérard, P. (2007). *Anticipatory Behavior in Adaptive Learning Systems*. Springer.

Carlsson, K., Petrovic, P., Skare, S., Petersson, K., and Ingvar, M. (2000). Tickling expectations: neural processing in anticipation of a sensory stimulus. *Journal of Cognitive Neuroscience*, 12(4):691–703.

Dayan, P. and Hinton, G. (1993). Feudal reinforcement learning. *Advances in Neural Information Processing Systems*, pages 271–271.

Gilbert, D. (2006). *Stumbling on Happiness*. Knopf Press.

Grush, R. (2004). The emulation theory of representation: motor control, imagery, and perception. *Behavioural and Brain Sciences*.

Hawkins, J. and Blakeslee, S. (2004). *On Intelligence*. Times Books.

Huron, D. (2006). *Sweet anticipation: Music and the Psychology of Expectation*. MIT Press.

Kaelbling, L. (1993). Learning to achieve goals. *In Proc. of Int. Joint Conf. on Artificial Intelligence (IJCAI-93)*.

Levitin, D. (2006). *This is Your Brain on Music*. Dutton Books.

Pavlov, I. (1927). *Conditioned Reflexes: An Investigations of the Physiological Activity of the Cerebral Cortex. (Translated and Edited by G. V. Anrep)*. Oxford University Press.

Pezzulo, G. (2008). Coordinating with the future: The anticipatory nature of representation. *Minds and Machines*, 18(2):179–225.

Rescorla, R. (1980). Simultaneous and successive associations in sensory preconditioning. *Journal of Experimental Psychology: Animal Behavior Processes*, 6(3):207–216.

Singh, S. (1992). Reinforcement learning with a hierarchy of abstract models. *Proc. Nat. Conf. on Artificial Intelligence (AAAI-92)*.

Sutton, R. S. (1995). TD models: Modeling the world at a mixture of time scales. *Proc. Int. Conf. on Machine Learning (ICML-95)*.

Sutton, R. S. and Barto, A. G. (1990). Time-Derivative Models of Pavlovian Reinforcement, In *Learning and Computational Neuroscience: Foundations of Adaptive Networks*, pages 497–537. MIT Press.

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. *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

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

Tolman, E. C. (1951). *Purposive Behavior in Animals and Men*. University of California Press.

Wolpert, D., Ghahramani, Z., and Jordan, M. (1995). An internal model for sensorimotor integration. *Science*, 269(5232):1880.