* Present a robot that learns to next in RT making thousand of predictions about sensory input signals at timescales from 0.1 to 8 seconds.
* Formulate prediction as GVF → the func. of sensory input signals is used as pseudo reward.
* Multi-time scale nexting in real time using just single tile coded feature representation.

Adaptive Behavior
2014, Vol. 22(2): 146–160

DOI: 10.1177/1059712313511648
adb.sagepub.com

Multi-timescale Nexting in a Reinforcement Learning Robot

→ the continual process of predicting what is likely to happen next.

* Nexting → personal and related to individual only
→ An individual may be making massive predictions (small) in parallel.

Joseph Modayil, Adam White, and Richard S. Sutton
Reinforcement Learning and Artificial Intelligence Laboratory
University of Alberta, Edmonton, Alberta, Canada

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, making thousands of predictions about sensory input signals at timescales from 0.1 to 8 seconds. Our predictions are formulated as a generalization of the value functions commonly used in reinforcement learning, where now an arbitrary function of the sensory input signals is used as a pseudo reward, and the discount rate determines the timescale. We show that six thousand predictions, each computed as a function of six thousand features of the state, can be learned and updated online ten times per second on a laptop computer, using the standard TD(λ) algorithm with linear function approximation. This approach is sufficiently computationally efficient to be used for real-time learning on the robot and sufficiently data efficient to achieve substantial accuracy within 30 minutes. Moreover, a single tile-coded feature representation suffices to accurately predict many different signals over a significant range of timescales. We also extend nexting beyond simple timescales by letting the discount rate be a function of the state and show that nexting predictions of this more general form can also be learned with substantial accuracy. General nexting provides a simple yet powerful mechanism for a robot to acquire predictive knowledge of the dynamics of its environment.

1. Multi-timescale Nexting

Psychologists have noted that people and other animals seem to continually make large numbers of short-term predictions about their sensory input (Gilbert 2006, Brogden 1939, Pezzulo 2008, Carlsson et al. 2000). When we hear a melody we predict what the next note will be or when the next downbeat will occur, and we 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

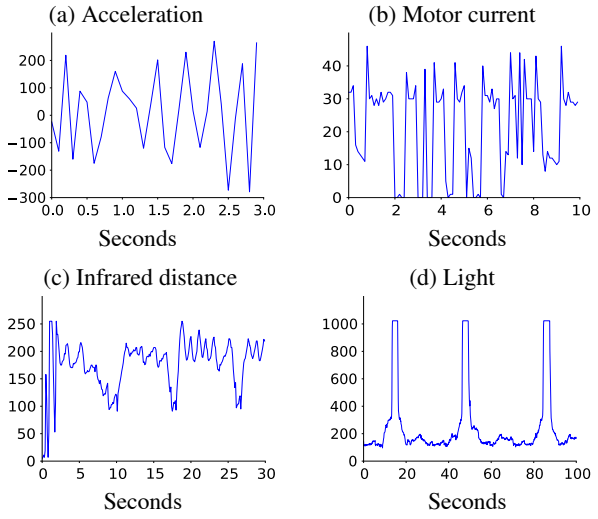

Fig. 1. Examples of sensory signals varying over very different time scales on the robot: (a) acceleration varying over tenths of a second, (b) motor current varying over fractions of a second, (c) infrared distance varying over seconds, and (d) ambient light varying over tens of seconds. The ranges of the sensor readings vary across the different sensor types.

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. In a similar fashion to these regularities in a person's experience, our robot has predictable regularities at time scales ranging from tenths of seconds to tens of seconds (Figure 1).

The ability to predict and anticipate has often been proposed as a key part of intelligence (e.g., Tolman 1951, Hawkins & Blakeslee 2004, Butz, Sigaud & Gérard 2003, Wolpert, Ghahramani & Jordan 1995, Clark 2013). 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 is the standard modern interpretation of the basic learning phenomenon known as *classical conditioning* (Rescorla 1980, Pavlov 1927). In a standard classical conditioning experiment, an animal is repeatedly given a sensory cue followed by a special stimulus that elicits a reflex response. For example, the sound of a bell might be followed by a shock to the paw, which causes limb retraction. The phenomenon is that after a while the limb starts to be retracted early, in response to the bell. This is interpreted as the bell causing a prediction of the shock, which then triggers limb retraction. In other experiments, for example those known as *sensory preconditioning* (Brogden 1939, Rescorla 1980), it has been 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 continually, but not expressed in behavior until some later experimental manipulation connects them to a response. Animals seem to 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 predictions—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, Ghahramani & Jordan 1995, Grush 2004, Sutton 1990). In this approach a small number of carefully chosen predictions are made of selected state variables. 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, Precup & Singh 1999, Dayan & 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.

The main contribution of this paper is a demonstration that many nexting predictions can be learned in real-time on a robot through the use of temporal difference methods. Our results show that learning thousands of predictions in parallel is feasible using a single representation and a single set of learning parameters. The results also demonstrate that the predictions achieve substantial accuracy within 30 minutes. Moreover, we show how simple extensions to the standard algorithm can express substantially more general forms of prediction, and predictions of this more general form can also be learned both accurately and quickly.

2. Nexting as Multiple Value Functions

We take a reinforcement-learning approach to achieving nexting. In reinforcement learning it is commonplace to use TD methods such as the TD(λ) algorithm (Sutton 1988) to learn long-term predictions of reward, called *value functions*. The TD(λ) algorithm has also been used as a model of classical conditioning (Sutton & Barto, 1990) within which various different stimuli are viewed as playing the role of reward in the learning algorithm. Our approach to nexting can be seen as taking this approach to the extreme, using TD(λ) to predict massive numbers of a great variety of reward-like signals at many time scales (cf. Sutton 1995, Sutton & Tanner 2005, Sutton et al. 2011).

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 0.1 seconds of real time. In conventional reinforcement learning, a single prediction is learned about a special signal called the "reward" and whose value at time t may be denoted $R_t \in \Re$. Here, we consider many predictions about many different signals. These signals are not goals in any sense, but they play the same role in the prediction-learning algorithm as reward; we call them *pseudo rewards*. The value at time t of the pseudo reward pertaining to the ith prediction is denoted $R_t^i \in \Re$. The prediction itself, denoted $V_t^i \in \Re$, is meant to approximate the discounted sum of the future values of the corresponding pseudo reward:

$$V_t^i \approx \sum_{k=0}^{\infty} (\gamma^i)^k R_{t+k+1}^i \stackrel{\text{def}}{=} G_t^i, \quad (1)$$

where $\gamma^i \in [0, 1)$ is the *discount rate* for the ith prediction. The discount rate determines the timescale of the prediction: to obtain a timescale of T time steps, the discount rate is set to $\gamma^i = 1 - \frac{1}{T}$. Readers familiar with reinforcement learning will recognize (1) as analogous to the definition of a state-value function. The prediction at time t is analogous to the approximated value of the state at time t, and G_t^i is analogous to the "return at time t" in reinforcement learning terminology. In this paper, G_t^i is the ideal value for the ith prediction at time t, and we refer to it as the *ideal prediction*. In our main experimental results, the pseudo reward was either a raw sensory signal or else a component of a state-feature vector (which we will introduce shortly), and the discount rate was one of four fixed values, $\gamma^i = 0$, 0.8, 0.95, or 0.9875, corresponding to timescales (T values) of 0.1, 0.5, 2, or 8 seconds.

We use linear function approximation to form each prediction. That is, we assume that the state of the world at time t is characterized by a feature vector $\phi_t \in \Re^n$ and that all predictions V_t^i are formed as inner products of ϕ_t with a corresponding weight vector θ_t^i:

$$V_t^i = \phi_t^\top \theta_t^i \stackrel{\text{def}}{=} \sum_{j=1}^{n} \phi_t(j)\theta_t^i(j), \quad (2)$$

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. In our experiments the feature vectors had $n = 6065$ components, but only a fraction of them were nonzero, so the sums could be very cheaply computed.

The predictions at each time are determined by the weight vectors θ_t^i. One natural and computationally frugal algorithm for learning the weight vectors is linear TD(λ), in which a small increment is made to each vector on each time step:

$$\theta_{t+1}^i = \theta_t^i + \alpha \left(R_{t+1}^i + \gamma^i \phi_{t+1}^\top \theta_t^i - \phi_t^\top \theta_t^i\right) z_t^i, \quad (3)$$

where $\alpha > 0$ is a step-size parameter, and $z_t^i \in \Re^n$, known as the *eligibility trace* vector, is initially set to zero and then updated on each step by

$$z_t^i = \gamma^i \lambda z_{t-1}^i + \phi_t, \quad (4)$$

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

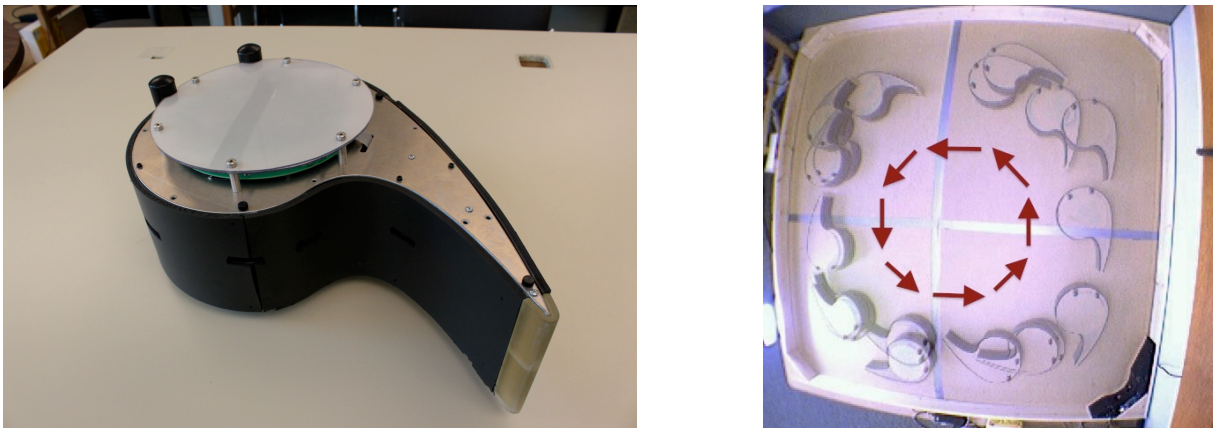

Fig. 2. The robot used in all experiments (left) and an illustration of its wall-following behavior (right) that generated the data. The circuits around the pen involved substantial random variation, but almost always included passing the bright light on the lower-left side.

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 the ideal prediction (1). In practice, smaller values of λ are almost always used because they can result in significantly faster learning (e.g., see Sutton & Barto 1998, Figure 8.15), but the $\lambda = 1$ case still provides an important theoretical touchstone. In this case we can define the *best static weight vector* θ^i_* as that which minimizes the squared error over the first N predictions:

$$\theta^i_* = \arg\min_\theta \sum_{t=1}^{N} \left(\phi_t^\top \theta - G^i_t\right)^2. \tag{5}$$

The best static weight vector can be computed offline by standard algorithms for solving large least-squares regression problems. The standard algorithm is $O(n^2)$ in memory and either $O(n^3)$ or $O(Nn^2)$ in computation, per prediction, and is just barely tractable for offline use at the scale we consider here (in which $n = 6065$). Although this algorithm is not practical for online use, its solution θ^i_* provides a useful performance standard. Note that even the best static weight vector will incur some error. It is even theoretically possible that an online learning algorithm could perform better than θ^i_*, by adapting to gradual changes in the world or robot. But under normal circumstances an online learning algorithm will only hope to approach the performance of the best static weight vector in the limit of infinite data.

Note that in presenting algorithms in this section we have carefully avoided any mention of expectations or states. We have written only of pseudo reward signals and feature vectors that can be directly computed from sensor readings. The algorithms are all well defined (and their performance can be assessed) even though conventional expectations and probabilities are not.

3. Scaling Experiment

We explored the practicality of applying computational nexting as described above to make and learn thousands of predictions, from thousands of features, in real time. We used a small mobile robot platform custom designed in our laboratory (Figure 2, left). The robot's primary actuators were three wheels placed in a standard omni-drive configuration enabling it to rotate and move in any direction. Sensors attached to the motors reported the electrical current, voltage, motor temperature, wheel rotational velocity, and an overheating flag, providing substantial observability of the internal physical state of the robot. Other sensors collected information from the external environment. Passive sensors detected ambient light in several directions in the visible and infrared spectrum. Active sensors on the sides of the robot emitted infrared (IR) light and measured the amount of reflected IR light, providing information about the distance to nearby obstacles. Other sensors measured acceleration, rotation, and the magnetic field. All together, the state of the robot was characterized by 53 real or virtual sensors of 13 types, as summarized in the first two columns of Table 1.

Sensor type	Num of sensors	tiling type	Num of intervals	Num of tilings
IRdistance	10	1D	8	8
		1D	2	4
		2D	4	4
		2D+1	4	4
Light	4	1D	4	8
		2D	4	1
IRlight	8	1D	8	6
		1D	4	1
		2D	8	1
		2D+1	8	1
Thermal	4(8)	1D	8	4
RotationalVelocity	1	1D	8	8
Magnetic	3	1D	8	8
Acceleration	3	1D	8	8
MotorSpeed	3	1D	8	4
		2D	8	8
MotorVoltage	3	1D	8	2
MotorCurrent	3	1D	8	2
MotorTemperature	3	1D	4	4
LastMotorRequest	3	1D	6	4
OverheatingFlag	1	1D	2	4

Table 1. Summary of the tile-coding strategy used to produce feature vectors from sensory signals. For each sensor of a given type, its tilings were either 1-dimensional or 2-dimensional, with the given number of intervals (see text and Figure 3). Only the first four of the robot's eight thermal sensors were included in the tile coding due to a coding error.

The robot's interaction with its environment was structured in a tight loop with a 100 millisecond (ms) time step. At each step, the sensory information was used to select one of seven actions corresponding to basic movements of the robot (forward, backward, slide right, slide left, turn right, turn left, and stop). Each action caused a different set of voltage commands to be sent to the three motors driving the wheels.

The experiment was conducted in a square wooden pen, approximately two meters on a side, with a lamp on one edge (Figure 2, right). The robot selected actions according to a fixed stochastic policy that caused it to generally follow a wall on its right side. The policy selected the forward action by default, the slide-left or slide-right action when the right-side-facing IR distance sensor exceeded or fell below given thresholds, and selected the backward action when the front IR distance sensor exceeded another threshold (indicating an obstacle ahead). We chose the thresholds such that the robot rarely collided with the wall and rarely strayed more than half a meter from the wall. By design, the backward action also caused the robot to turn slightly to the left, facilitating the many left turns needed for wall following on the right. To inject some variability into the behavior, on 5% of the time steps the policy instead chose an action at random from the seven possibilities with equal probability. Following this policy, the robot usually completed a circuit of the pen in about 40 seconds. A circuit took significantly longer if the motors overheated and temporarily shut themselves down. In this case the robot did not move, irrespective of the action chosen by the policy. Shut downs occurred approximately every 8 minutes and lasted for about 7 minutes. This simple policy was sufficient for the robot to reliably follow the wall for hours.

To produce the feature vectors needed for the TD(λ) algorithm, the sensor readings were coarsely coded as 6065 binary features according to a tile-coding strategy as summarized in Table 1 and exemplified in Figure 3. Tile coding is a standard technique for converting continuous variables into a sparse feature representation that is well suited for online learning algorithms. The sensor readings are taken in small groups and partitioned, or tiled, into non-overlapping regions called tiles. One such tiling over two sensor readings from our robot is shown on the left side of Figure 3. In this case the tiling is a simple regular grid of square tiles of equal width (for some other possibilities see Sutton & Barto 1998, p. 206-7).

Tile coding becomes much more powerful than a simple discretizing of the state space through the use of multiple overlapping tilings that are offset from each other as shown in the right side of Figure 3. For each tiling, a given state (e.g., the state marked by a white dot in the figure) is in exactly one tile. The set of tiles that are activated by a state constitute a coarse coding of the state's location in sensor space. The resolution of this coding is finer than that of the individual tilings, as suggested by the greater density of lines in Figure 3 (right). With four tilings, the effective resolution is roughly four times that of the original tiling. The advantage of the multiple tilings over a single tiling with four times the resolution is that generalization will be broader with multiple tilings, which typically leads to much faster learning. With tile coding one can quickly learn a coarse approximation to the desired mapping, and then refine it with further data, simultaneously obtaining the benefits of both coarse and fine discretizations.

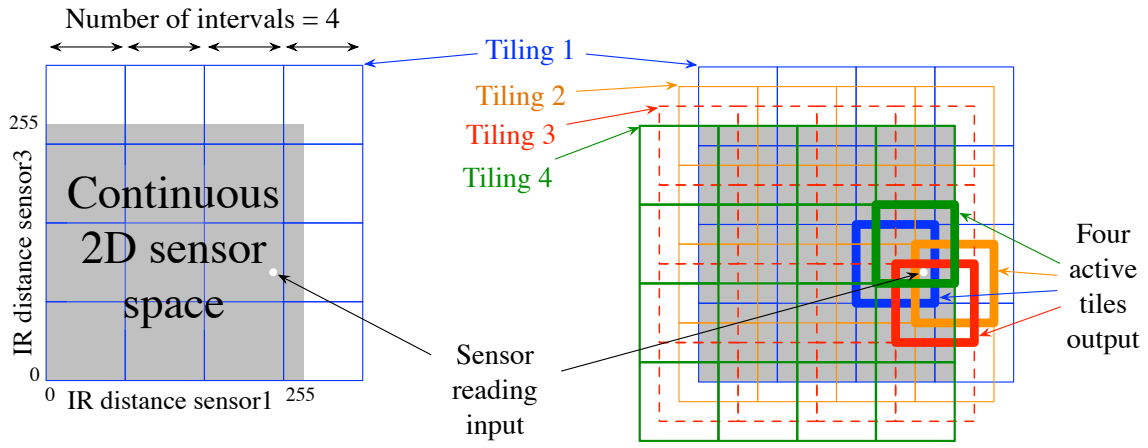

Fig. 3. An example of how tile coding was used to map continuous sensor input into many binary features. On the left we see a single tiling of the continuous 2D space corresponding to the readings from two non-consecutive (2D+1) IR distance sensors. The space was tiled into 4 intervals in each dimension, for 16 tiles overall. On the right we see all four tilings, each offset by a different negative amount such that they were equally spaced, with the first tiling starting at the lower left of the sensor space (as shown on the left) and the last tiling ending at the upper right of the space. A sensor reading input to the tile coder is a point in the space, like that shown by the white dot. The output of tile coding is the indices of the four tiles that contain the point, as shown on the right. These tiles are said to be active, and their corresponding features take on the value 1, while all the non-active tiles correspond to features with the value 0. Note how the four tilings provide a dense grid of lines, each a distinction that can be made between input points, yet the four active tiles together span a substantial portion of the sensor space. In this way, multiple tilings provides a feature representation that enables both fine resolution *and* broad generalization. This tile-coding example corresponds to the fourth row of Table 1.

Each tile in a tiling corresponds to a single binary feature. If the current sensor readings fall in that tile, then the feature is active and takes the value 1, otherwise it is inactive and takes the value 0. In our representation, we had $n = 6065$ such features making up our binary feature vectors, $\phi_t \in \{0,1\}^{6065}$. The specifics of our tile-coding strategy are summarized in Table 1. Most of our tilings were 1-dimensional (1D), that is, over a single sensor reading, in which case a tile was simply an interval of the sensor reading's value. For some sensors we formed 2-dimensional (2D) tilings by taking neighboring sensors in pairs. This enabled the robot's features to distinguish between, for example, a wall and a corner. To provide further discriminatory power, for some sensor types we also formed 2-dimensional tilings from pairs consisting of a sensor and its second-neighboring sensor. These are indicated as 2D+1 tilings in Table 1 (and this is the specific case illustrated in Figure 3). Finally, we added a tiling with a single tile that covered the entire sensor space and thus whose corresponding feature, called the *bias feature*, was always active. Altogether, our tile-coding strategy used 457 tilings, producing feature vectors with $n = 6065$ components, most of which were zeros, but exactly 457 of which were ones.

We first applied TD(λ) to make and learn 2160 predictions. The pseudo-reward signals R_t^i of the predictions were the 53 sensor readings and a random selection of 487 from the 6064 non-bias features. For each of these signals, four predictions were learned with the four values of the discount rate, $\gamma^i = 0, 0.8, 0.95$, and 0.9875, corresponding to timescales of 0.1, 0.5, 2, and 8 seconds respectively. Thus, we sought to learn a total of $(53 + 487) \times 4 = 2160$ predictions. The learning parameters were $\lambda = 0.9$ and $\alpha = \frac{0.1}{457}$ (as there are 457 active features), and the initial weight vector was zero. Data was logged to disk for later analysis. The total run time for this experiment was approximately three hours and twenty minutes (120,000 time steps).

We can now address the main question: is real-time nexting practical at this scale? In our setup, this comes down to whether or not all the computations for making and learning so many complex predictions can be reliably completed within the robot's 100ms time step. The wall-following policy, tile-coding, and TD(λ) were all implemented in Java and run on a laptop computer 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

53 sensors
487 random selection from 6064 fts.
(53+487) x 4 → discount factors
= 2160

* TD(λ) → applied to make & learn 2160 predictions
Pseudo-reward signals → R_t^i of the predictions were 53 sensor readings and a random selection of 487 from 6064 non-bias features.

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. The total memory consumption was 400MB. With this setup, the time required to make and update all 2160 predictions was 55ms, well within the 100ms duty cycle of the robot. This demonstrates that it is indeed practical to do large-scale nexting on a robot with conventional computational resources.

Later, on a newer laptop computer (Intel Core i7, 2.7 Ghz quad core, 8GB 1600 Mhz DDR3 RAM, 8 threads), with the same style of predictions and the same features, we found that we were able to make 6000 predictions in 85ms. This shows that with more computational resources, the number of predictions (or the size of the feature vectors) can be increased proportionally. This strategy for nexting easily scales to millions of predictions with foreseeable increases in computing power over the next decade.

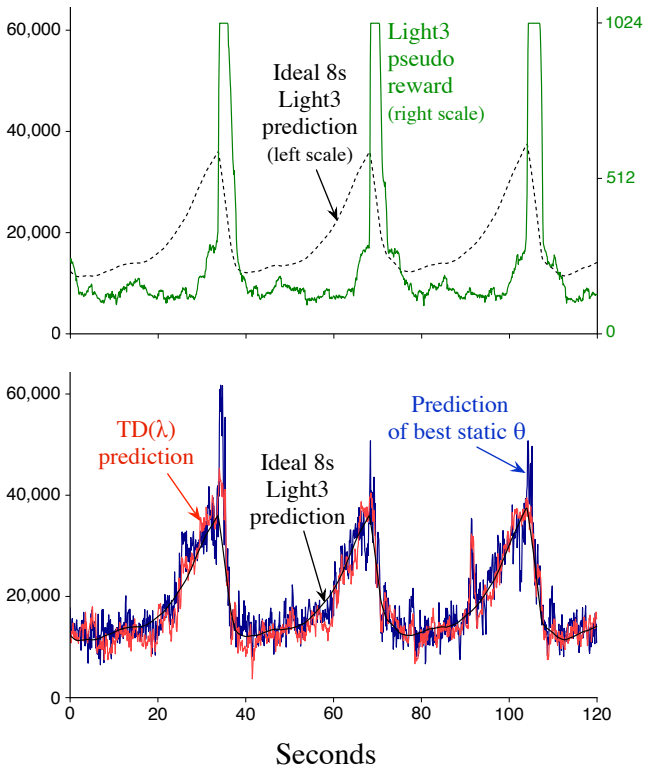

Fig. 4. Predictions of the Light3 pseudo reward at the eight-second timescale. The upper graph shows the Light3 sensor reading spiking and saturating on three circuits around the pen and the corresponding ideal prediction (computed afterwards from the future pseudo rewards). Note that the ideal prediction shows the signature of nexting—a substantial increase prior to the spikes in pseudo reward. The lower graph shows the same ideal prediction compared to the prediction of the TD(λ) algorithm and of the prediction of the best static weight vector. These feature-based predictions are more variable, but substantially track the ideal.

4. Accuracy of Learned Predictions

The predictions were learned with substantial accuracy. For example, consider the eight-second prediction whose pseudo reward is the third light-sensor reading (Light3). Notice that there is a bright lamp in the lower-left corner of the pen (Figure 2, right). On each trip around the pen, the reading from this light sensor increased to its maximal level and then fell back to a low level, as shown in the upper portion of Figure 4. If the state features are sufficiently informative, then the robot should be able to anticipate the rising and falling of this sensor reading. Also shown in the figure is the ideal prediction for this time series, G_t^i, computed retrospectively from the subsequent readings of the light sensor. Of course no real predictor could achieve this—our learning algorithms seek to approximate this 'clairvoyant' prediction using only the sensory information available in the current feature vector.

The prediction made by TD(λ) is shown in the lower portion of Figure 4, along with the prediction made by the best static weight vector θ_*^i computed retrospectively as described in Section 2. The key result is that the TD(λ) prediction anticipates both the rise and fall of the light. Both the learned prediction and the best static prediction track the ideal prediction, though with visible fluctuations.

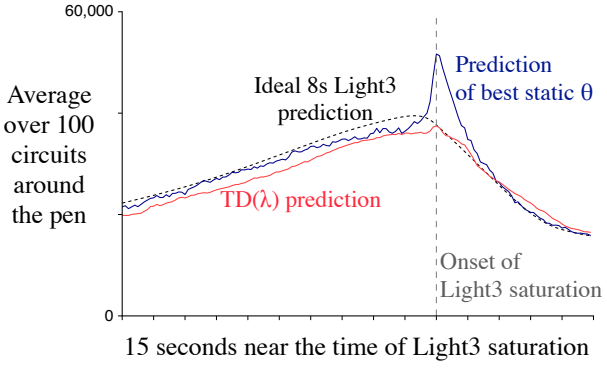

Fig. 5. Light3 predictions (like those in the lower portion of Figure 4) averaged over 100 circuits around the pen and aligned at the onset of Light3 saturation.

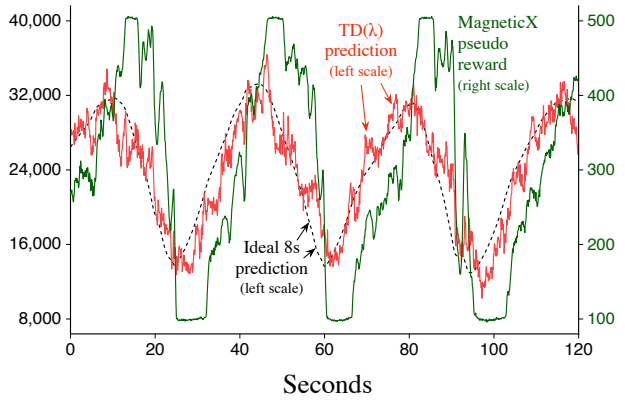

Fig. 6. Predictions of the MagneticX sensor at the eight-second timescale. The TD(λ) prediction was close to the ideal prediction, explaining 90 percent of its variance.

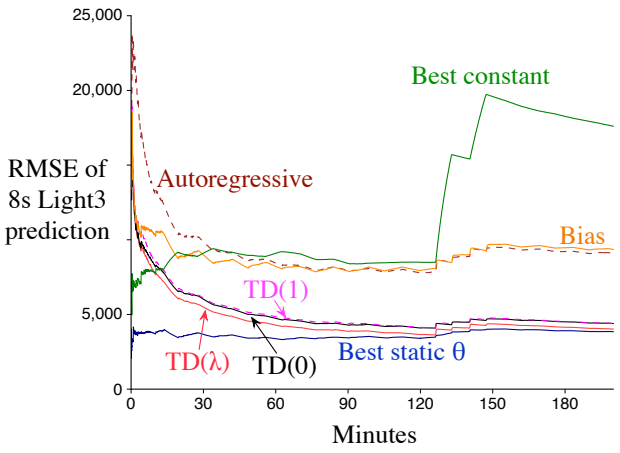

Fig. 7. Learning curves for 8-second Light3 predictions made by various algorithms over the full data set. Each point is the RMSE of the prediction of the algorithm up to that time. Most algorithms use only the data available up to that time, but the best-static-θ and best-constant algorithms use knowledge of the whole data set. The errors of all algorithms increased at about 130 and 150 minutes because the motors overheated and shutdown at those times while the robot was passing near the light, causing an unusual pattern of sensor readings. In spite of the unusual events, the RMSE of TD(λ) still approached that of the best static weight vector. See text for the other algorithms.

To remove these fluctuations and highlight the general trends in the eight-second predictions of Light3, we averaged the predictions over 100 circuits around the pen, aligning each circuit's data to the time of initial saturation of the light sensor. The average of the ideal, TD(λ), and best-static-weight-vector predictions for 15 seconds near the time of saturation are shown in Figure 5. All three averages rise in anticipation of the onset of Light3 saturation and fall rapidly afterwards. The ideal prediction peaks before saturation, because the Light3 reading regularly became elevated prior to saturation. The two learned predictions are roughly similar to the ideal, and to each other, but there are substantial differences. These differences do not necessarily indicate error or essential characteristics of the algorithms. For example, such differences can arise because the average is over a biased sample of data—those time steps that preceded a large rise in the pseudo reward. We have established that some of the differences are due to the motor shutdowns. Notably, if the data from the shutdowns are excluded, then the prominent bump in the best-static-θ prediction (in Figure 5) at the time of saturation onset disappears.

Figure 6 shows another example of the accuracy of the near-final TD(λ) predictions, in this case of one of the Magnetometer sensor readings at an eight-second timescale.

We turn now to consider how the quality of the eight-second Light3 prediction evolves over time and data. As a measure of the quality of a prediction sequence $\{V_t^i\}$ up through time T, we use the root mean squared error, defined as:

$$\text{RMSE}(i,T) = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(V_t^i - G_t^i)^2}.$$

Figure 7 shows the RMSE of the eight-second Light3 predictions for various algorithms. For TD(λ), the parameters were set as described above. For all the other algorithms, their parameters were tuned manually to optimize the final RMSE. The other algorithms included TD(λ) for $\lambda = 0$ and $\lambda = 1$, both of which performed slightly worse than $\lambda = 0.9$. Also shown is the RMSE of the prediction of the best static weight vector and of the best constant prediction. In these cases the prediction function does not actually change over time, but the RMSE measure varies as harder or easier states from which to make predictions are encountered. Note that the RMSE of the TD(λ) prediction comes to closely approach that of the best static weight vector after about 90 minutes. This demonstrates that online learning on robots can be effective in real time with a few hours of experience, even with a large state representation.

The benefits of a large representation are shown in Figure 7 by the substantially improved performance over the 'Bias'

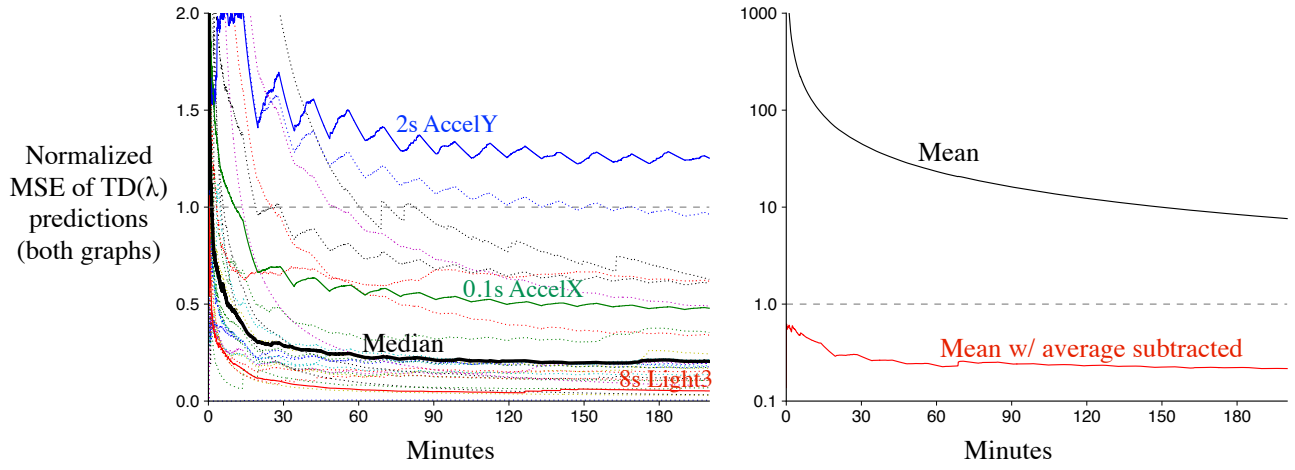

Fig. 8. Learning curves for the 212 predictions whose pseudo reward is a sensor reading. The median and several representative learning curves are shown on a linear scale on the left, and the mean learning curve is shown on a logarithmic scale on the right. The mean curve is high because of a minority of the sensors whose absolute values are high and whose variance is low. If the experiment is rerun using pseudo rewards with their average value subtracted out, then the mean performance is greatly improved, as shown on the right, explaining 78% of the variance in the ideal prediction by the end of the data set.

algorithm, which was TD(0) with a trivial representation consisting only of the bias feature (the single feature that is always 1). As an additional performance standard, also shown is the RMSE of an autoregressive algorithm (e.g., see Box, Jenkins & Reinsel 2011) that uses previous readings of the Light3 sensor as features of a linear predictor, with the weights trained according to the least-mean-square rule. To incrementally train the autoregressive model, the learning was delayed by 600 timesteps to compute the ideal prediction. The best performance of this algorithm was obtained using a model of order 300, meaning the last 300 readings of the Light3 sensor were used. The autoregressive model performed much worse than all the algorithms that used a rich feature representation.

Moving beyond the single prediction of one light sensor at one timescale, we next evaluate the accuracy of all 212 predictions about sensors at various timescales. To measure the accuracy of predictions with different magnitudes, we used a normalized mean squared error,

$$\text{NMSE}(i,t) = \frac{\text{RMSE}^2(i,t)}{\text{var}(i)},$$

in which the mean squared error is scaled by $\text{var}(i)$, the sample variance of the ideal predictions G_t^i over all the timesteps. This error measure can be interpreted as the percent of variance not explained by the prediction. It is equal to one when the prediction is constant at the average ideal prediction.

Learning curves using the NMSE measure for the 212 predictions whose pseudo reward was a sensor reading are shown in Figure 8. The left panel shows the median learning curve and curves for a selection of individual predictions. In most cases, the error decreased rapidly over time, falling substantially below the unit variance line. The median prediction explained 80% of the variance at the end of training and 71% of the variance after just 30 minutes. In many cases the decrease in error was not monotonic, sometimes rising sharply (presumably as a new part of the state space was encountered) before falling further. In some cases, such as the 2-second Y-acceleration prediction shown, the sensor was never effectively predicted, evidenced by its NMSE never falling below one. We believe this signal was simply unpredictable with the feature representation provided.

The mean learning curve, shown in the right panel of Figure 8 on a log scale, fell rapidly but was always substantially above one. This was due to a minority of the sensors (mainly the thermal sensors) whose values were far from zero but whose variance was small. The learning curves for the corresponding predictions were all very high (and do not appear in the left panel because they were way off the scale). Why did this happen? Note that our prediction algorithm was biased in that all the initial predictions were

zero (because the weight vector was initialized to zero). When the pseudo rewards are large relative to their variance, this bias can result in a very large NMSE that takes a long time to subside. One way to eliminate the bias is to modify the pseudo rewards by subtracting from each sensor value the average of its values up to that time (e.g., the first pseudo reward is always zero). This is easily computed and uses only information readily available at the time. Most importantly, choosing the initial predictions to be zero is no longer a bias but simply the right choice. When we modified our pseudo rewards in this way, and reran TD(λ) on the logged data, we obtained the much lower mean learning curve shown in Figure 8 (right). In the mean, the prediction learned with the average subtracted explained 78% of the variance of the ideal prediction by the end of the data set.

Finally, consider the majority of the predictions whose pseudo reward was one of the binary features making up the feature vector. There were 170 constant features among the 487 binary features that were selected to be pseudo rewards, and thus with the average subtracted, both the ideal predictions and the learned predictions were constant at zero. For these constant predictions, the RMSE was zero and the variance was zero, and we excluded these predictions from further analysis. For the remainder of the predictions, both the median and mean explained 30% of the variance of the ideal prediction by the end of the data set.

These results provide evidence that real-time parallel learning of thousands of accurate nexting predictions on a physical robot is possible and practical. Learning to substantial accuracy was achieved within 30 minutes of training, with no tuning of algorithm parameters, and using a single feature representation for all predictions. The parallel scalability of knowledge-acquisition in this approach is substantially novel when compared with the predominately sequential existing approaches common for robot learning. Our results also show that online methods can be competitive in accuracy with an offline optimization method.

5. Beyond Simple Timescales

In this section we present a small generalization of the TD(λ) algorithm that enables it to learn predictions of a significantly more general and expressive form. Up to now, the discount rate, γ^i, has been varied only from prediction to prediction; for the ith prediction, γ^i was constant and determined its timescale. Now we will allow the discount rate for an individual prediction to vary over time depending on the state the robot finds itself in; we will denote its value at time t as $\gamma^i_t \in [0,1]$. With a constant discount rate, predictions are restricted to simple timescales in which pseudo rewards are weighted geometrically less the more they are delayed, as was given by the earlier definition of the ideal prediction:

$$V^i_t \approx \sum_{k=0}^{\infty} (\gamma^i)^k R^i_{t+k+1} \stackrel{\text{def}}{=} G^i_t. \tag{1}$$

With a variable discount rate, the weighting is not by simple powers of γ^i, but by *products* of γ^i_t:

$$V^i_t \approx \sum_{k=0}^{\infty} \left(\Pi_{j=1}^{k} \gamma^i_{t+j}\right) R^i_{t+k+1} \stackrel{\text{def}}{=} G^i_t. \tag{6}$$

The learning algorithm remains unchanged in form and computational complexity; it is exactly as given earlier, except with γ^i replaced by γ^i_t or γ^i_{t+1}, as appropriate:

$$\theta^i_{t+1} = \theta^i_t + \alpha \left(R^i_{t+1} + \gamma^i_{t+1}\phi^\top_{t+1}\theta^i_t - \phi^\top_t \theta^i_t\right) z^i_t, \tag{7}$$

$$z^i_t = \gamma^i_t \lambda z^i_{t-1} + \phi_t. \tag{8}$$

This small change results in a significant increase in the kinds of ideal predictions that can be expressed (Sutton 1995, Maei & Sutton 2010, Sutton et al. 2011). Our contribution in this section is to apply and demonstrate this generalization of TD(λ) in three examples of nexting in robots.

For the first example, consider a discount rate that is usually constant and near one, but falls to zero when some designated event occurs. In particular, consider

$$\gamma^i_t = \begin{cases} 0 & \text{if Light3 is saturated at time } t; \\ 0.9875 & \text{otherwise.} \end{cases}$$

As long as Light3 is not saturated, this discount rate works like an ordinary eight-second timescale—pseudo rewards are weighted by 0.9875 carried to the power of how many steps they are delayed. But if Light3 ever becomes saturated, then all pseudo rewards after that time are given zero weight. This kind of discount enables us to predict how

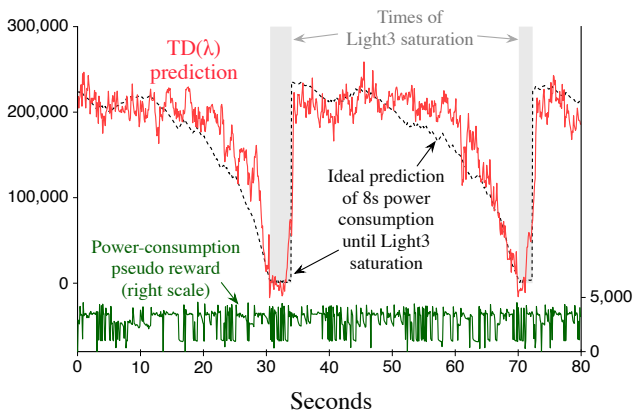

Fig. 9. Predictions of total power consumption, over an eight-second time scale, or up until the Light3 sensor reading is saturated. To express this kind of prediction, the discount rate must vary with time (in this case dropping to zero upon Light3 saturation).

much of something will occur prior to a designated event (in this case, prior to Light3 saturation).

The pseudo reward in this example is a measure of the total power consumption of the three motors,

power consumption of three motors.

$$R_t^i = \sum_{j=1}^{3} |\text{MotorVoltage}j_t \times \text{MotorCurrent}j_t|.$$

As shown in Figure 9, power consumption tended to vary between 1000 and 3000 depending on how many motors were active. The ideal prediction, also shown in Figure 9, was similar to that of an eight-second prediction for much of the time, but notice how it falls all the way to zero during Light3 saturation. Even though there was substantial power consumption within the subsequent eight seconds, this has no effect on the ideal prediction because of the saturation-triggered discounting. The figure shows that the modified TD(λ) algorithm performed well here (after training on the previous 150 minutes of experience): over the entire data set the predictions captured approximately 88% of the variance in the ideal prediction.

The ideal prediction in the above example, like those of simple timescales, always weights delayed pseudo rewards less than immediate ones. It cannot put higher weight on the pseudo rewards received later than those received immediately. This limitation is inherent in the definition of the ideal prediction (6) together with the restriction of the discount rate to $[0, 1]$. However, it is only a limitation with respect to the pseudo reward; if signals are mixed into the pseudo reward in the right way, then predictions about the signals can be made with general temporal profiles. In particular, it may be useful to predict what value a signal will have at the time some event occurs. For example, suppose we have some signal X_t whose value we wish to predict not in the short term, but rather at the time of some event. To do this, we construct a discount rate γ_t^i that is one up until the event has occurred, then is zero. The pseudo reward is then constructed as follows:

$$R_t^i = (1 - \gamma_t^i) X_t. \tag{9}$$

This pseudo reward is forced to be zero prior to the event (because $1 - \gamma_t^i$ is zero) and thus nothing that happens during this time can affect the ideal prediction. The ideal prediction will be exactly the value of X_t at the time γ_t^i first becomes zero.

Constructing the pseudo reward by (9) has several possible interpretations depending on the exact form of X_t and γ_t^i. If X_t is the indicator function for an event (equal to one during it, zero otherwise), and γ_t^i is a constant less than one prior to the event (and zero during the event), then the prediction will be of how imminent the onset of the event is. Figure 10 shows results with an example of this using the data from our robot: γ_t^i prior to the event was 0.8 (corresponding to a half-second timescale), and the event was a right-facing IR sensor exceeding a threshold (corresponding to being within 12cm of the wall).

Our final example, in Figure 11, illustrates the use of signals X_t that are not binary and discount rates γ_t^i that do not fall all the way to zero. The idea here is to predict what the four light-sensor readings will be as the robot rounds the next corner of the pen. Four predictions are created, one for each light sensor, with signals $X_t^i = \text{Light}i$ equal to the sensor reading. Rounding a corner is an event indicated by a value from the side IR distance sensor corresponding to a large distance ($>\approx$25cm). This typically occurs for several seconds during the corner's turn. We set the discount rate γ_t^i equal to 0.9875 (an eight-second timescale) most of the time, and equal to 0.3 when rounding a corner. Because the discount rate is greater than zero during the event, the light readings from several time steps contribute to the ideal prediction as the corner is entered.

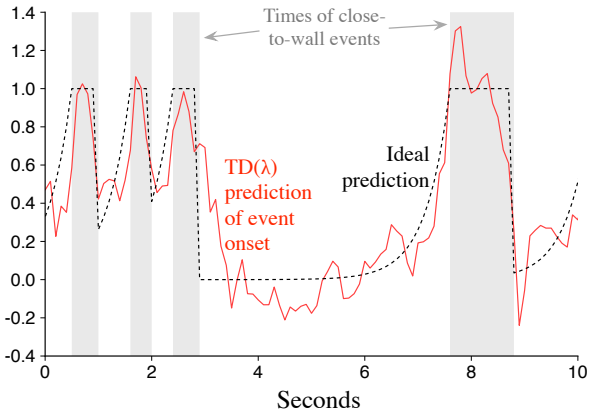

Fig. 10. Predictions of the imminence of the onset of an event regardless of its duration. The event here is being too close (<≈12cm) to a side wall of the pen, and imminence is with respect to a half-second timescale. The learned predictions rise before the event and follow the shape of the ideal predictions. Overall, the residual unexplained variance of this prediction was 23%.

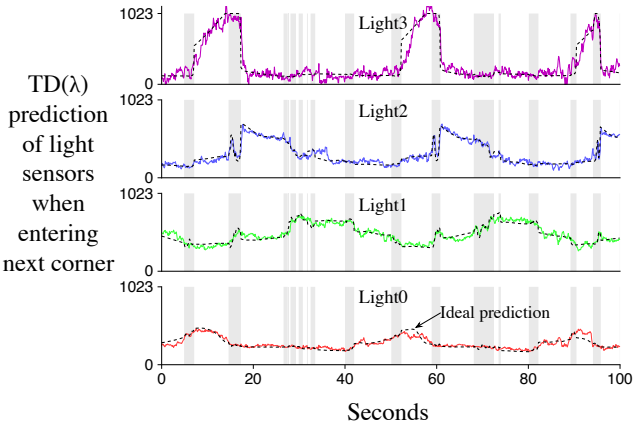

Fig. 11. Predictions of what each of the four light-sensor readings will be when the robot rounds its next corner. The greyed time steps indicate those in which the robot was considered to be rounding a corner. The residual unexplained variances for the four predictions were 6%, 10%, 10%, and 11% over the course of the data set.

6. Discussion

We have successfully implemented a robot version of the psychological phenomenon of nexting. The robot learned to predict thousands of aspects of its near future experience ten times each second. It predicted at a range of timescales, from one-tenth of a second to eight seconds, and also beyond simple timescales. Perhaps our most important result was to show that robot nexting is not only possible, but eminently practical. Using computationally inexpensive methods such as TD(λ), linear function approximation, and tile coding, we showed that the nexting computations easily scale to thousands of predictions based on thousands of features on a small computer. Although these algorithms are computationally cheap, they worked well. An extensive analysis of a subset of the learned predictions found them to be substantially accurate within 30 minutes of real-time training—fast enough for frequent retraining or adaptation to new sensors or environments. It is also notable that we used a single set of parameters and a single set of features for all predictions, despite variations in signal scales, signal variability, and timescales. Being able to treat all predictions uniformly in these ways facilitates the general application of nexting.

6.1. Relationship to conventional robotics

From the perspective of conventional robotics research, there are three aspects of our nexting robot that are distinctive.

The first is that the robot updates a very large number of predictions, in real time, about diverse aspects of its experience, giving it a distinctively rich awareness of its surroundings. This contrasts with the conventional approach to robot engineering, in which designers identify the minimal set of state variables needed to solve a specific task, and the robot is oblivious to all others. This approach is perhaps the source of the popular notion that to perform some task "like a robot" is to do it with minimal awareness and understanding.

Nowadays, it is not unusual for advanced robots to have a substantial awareness of their environment. The premier example of this is probably self-driving cars (e.g., Wang, Thorpe & Thrun 2003, see Markoff 2010). These systems can simultaneously track many objects including cars, people, bicycles, and traffic signs. Another important example is given by Simultaneous Localization And Mapping (SLAM) robots that construct occupancy maps for the space around the robot (Thrun, Montemerlo, et al. 2006). SLAM robots have considerable scale but do not predict a diversity of objects or sensor types. Both of these systems are

highly engineered and the things predicted by the underlying algorithms are highly interrelated and cover a few structured types. This contrasts with our approach to nexting, in which no prior model is used and each prediction is formed independently of the others. Whatever the other merits or drawbacks of our approach, it does enable easy scaling to large numbers of sensors of arbitrary types. Even on our small research robot, we demonstrated learning with a greater variety of sensor types than in any SLAM robot, and perhaps more even than in current self-driving cars.

The second way in which our nexting robot is unusual is that it learns to make its predictions *online*, during its normal operation. Most learning robots complete their learning before being put into use, in special training sessions requiring information that will not be available during use, such as human-provided labels, demonstrations, or calibrations. Most of the learning in self-driving cars and in SLAM robots is of this sort, with important final tuning and local mapping done online. Classical state-estimation methods such as Kalman filters adapt only low-dimensional gain parameters online. Finally, there have been a handful of works with reinforcement learning robots that learn value functions or policies online (e.g., Peters & Schaal 2008, Tedrake, Zhang & Seung 2005, see Degris et al. 2012). In all cases, the online learning is limited in its scale and diversity; it never approaches the adaptive awareness of our nexting robot with its online, ten-times-per-second learning of thousands of diverse predictions.

The third way in which our nexting robot is distinctive is that its predictions are relatively long-term, extending significantly beyond a single time step. Although prediction is widely used in modern control theory, it is almost always limited to one-step (or differential) predictions (e.g., conventional Kalman filtering (Welch & Bishop 1995) and system identification (Ljung 1998)). Often, one-step predictive models are iterated to make multi-step predictions (e.g., model-predictive control (Camacho & Bordons 2004) and motion planning (LaValle 2006)). That can work well, but it does not scale to long timescales or to large numbers of predictions such as we have used here. Another way of making longer-term predictions with one-step methods is to make the step larger, subsampling or jumping through the data stream at multiple temporal resolutions. A weakness of this approach is that the longer-term predictions are also made less frequently and are thus not available to affect a rapid response if needed. In addition, it seems unlikely that this approach could extend beyond simple timescales to the more general predictions described in the previous section. Why are all these techniques, and nexting itself, focused on constructing longer-term predictions? The advantage of predicting events substantially in advance of their occurrence is that it enables appropriate action to be taken to head off or otherwise prepare for them. In brief, long-term prediction is essential to *anticipation*, and thus to the timely generation of appropriate responses.

6.2. The distinctiveness of predictions in AI

Predictions are a potentially powerful organizing principle for artificial intelligence (AI) systems. Through our focus on nexting in this paper, we have explored one small way in which predictions can be important in an AI system. Even so, our nexting robot constitutes one of the most well-developed uses of predictions in AI research to date, as we discuss in this subsection.

Of the previous AI systems that have learned from a robot's sensorimotor experience, most have not expressed their knowledge in a predictive form and validated it by comparison with subsequent experience. Pierce and Kuipers (1997) gathered sensorimotor experience from random motion on a simulated robot, and then constructed a low dimensional embedding of the sensors from observed correlations between sensor readings. The validity of the embedding was assessed by how well the constructed embedding matched the spatial configuration of the robot's sensors provided by the experimenter. Oates, Schmill, and Cohen (2000) described an algorithm that segments time-series data from a robot's sensors and forms clusters from temporal segments with similar dynamics. The clustering was validated by how well these clusters matched clusters generated by people. Predictions were used by Yamashita and Tani (2008), but the predictions were constrained only be learning from people during a special supervisory training period. They taught a humanoid robot to perform goal-directed reaching motions in response to human-issued commands using predictions about how a person would move the robot within a supervisory training mode. Our work with nexting shows how predictions can be learned

from normal robot operation without the need for external validation. Predictions are a special form of sensorimotor knowledge for which future experience is both necessary and sufficient for validation.

Compared to previous AI research on predictive knowledge validated from experience, our work is distinctive in showing practicality and scalability with a physical robot. That knowledge might be expressed in terms of predictions has been explored by Cunningham (1972), Becker (1973), Drescher (1990), and Sutton (2009, 2012), but only in small scale simulations, in most cases with substantial abstractions given a priori. Our results show that a predictive approach is practical on a physical robot from the level of sensors and motors, using features that are constructed from the same. The required computation for making and updating thousands of predictions was provided by a laptop that can be carried by a small robot. Moreover, the predictions were learned with accuracy, with uniform parameter settings and within a few hours of experience. Our results demonstrate that a single mathematical form of expectation suffices for expressing these many different predictions, covering many sensors, features, and timescales, and that this form generalizes beyond timescales.

From the perspective of AI more generally, our approach is distinctive in pursuing knowledge representation empirically through a diverse set of predictions. In contrast to conventional approaches, we are abandoning the need for the robot's knowledge to be consistent or complete with respect to some human-constructed model of the world. Instead we formulate precise empirical predictions and then learn them from the robot's experience. This is a piecewise approach to knowledge that we see as part of respecting the complexity of the world. We expect that many robots will have neither the ability nor the need to model all aspects of the world, and can do well by predicting those aspects of the world that are manifest in their sensorimotor experience.

6.3. Uses of predictions

This paper has extensively treated the learning of nexting-style predictions without proposing specific ways that the predictions should be used. On balance we believe that this is appropriate; learning large number of predictions in real time is itself very challenging, and the potential uses of predictions are too many and too varied to treat properly in the same paper. Nevertheless, it is appropriate to at least briefly mention some of the possible uses of nexting-style predictions.

Suppose a self-driving car is using visual input to predict subsequent laser readings corresponding to obstacles or rough terrain. These predictions might be used to slow the vehicle in advance to avoid abrupt braking, a rough ride, or collisions. Such predictions could also be used to choose between alternate routes. As another example, a vacuuming robot could predict its remaining battery life and its time to return to its docking station to determine when to head back to recharge. Or, taking a cue from psychology and the biological examples of nexting, predictions could be used for anticipatory reflexes for self-preservation. Just as a rabbit closes its eye in anticipation of an oncoming irritant, a robot could cover its sensory surfaces, or spin down its hard disk, if it predicted that it might topple. Predictions would also be useful as part of anomaly detection, for example, in detecting unusual patterns of computer use that indicate an intruder.

In addition to these fairly obvious ways to use predictions, there are others that are more nuanced, with a less direct connection to behavior. One sophisticated use of predictions is to package them into *option models*, a special form of temporally abstract prediction specially suited to planning (Sutton, Precup & Singh 1999). Another nuanced use of predictions is as components of state representations, as in predictive state representations (Littman, Sutton & Singh 2002, Singh, James & Rudary 2004, Boots, Siddiqi & Gordon 2011, Sutton & Tanner 2005). The scale at which we have demonstrated nexting-style predictions suggests that practical benefits might be achieved from using predictions in such ways.

7. Limitations and Conclusion

The most important limitation of this work is that all of the predictions learned were conditional on the one way in which the robot behaved. In other words, they were all *on-policy* predictions rather than *off-policy* predictions. Off-policy learning adds substantial expressive power and is significantly more challenging (e.g., see Sutton, Szepesvári

& Maei 2009). Gradient-TD methods (Maei 2011, Sutton et al. 2009) have been developed to deal with the most serious challenges of off-policy learning, and Sutton et al. (2011) have already used them to demonstrate off-policy TD learning in robots on a small scale. These methods could probably be extended to real-time parallel learning of many predictions with modest increases in the computational resources (see White, Modayil & Sutton 2012). Perhaps the most important advantage of moving to off-policy prediction is that it frees us to choose the robot's behavior for other purposes. In particular, we may desire the robot's behavior to change over time, say to maximize either some extrinsic reward or the total amount of learning, as in work on computational curiosity (e.g., see Oudeyer, Kaplan & Hafner 2007).

We have demonstrated that the psychological phenomena of nexting—learning and making thousands of local, short-term, personal predictions—can be produced in a robot in a practical, scalable way using modern conventional computers. Our approach was to formulate the predictions in the form of value functions, like those conventionally used in reinforcement learning, but on a much larger scale. Our robot predicted 6000 aspects of 53 sensory signals at four timescales, with the predictions made and learned every tenth of a second. We used a large feature representation and the TD(λ) online learning algorithm. We showed that a single feature representation and a single set of learning parameters was sufficient for learning many diverse predictions in 30 minutes or less. Finally, we showed that a natural extension of our approach—allowing the discount rate of a prediction to vary with the current state instead of being constant—provides substantial additional flexibility and expressive power. Overall, our results suggest that nexting might form a competent starting place for developing a sensorimotor approach to artificial intelligence.

Acknowledgements

This paper was extended and revised from the work presented in Modayil, White & Sutton (2012). The authors thank Mike Sokolsky for creating the robot and thank Patrick Pilarski and Thomas Degris for assisting in the preparation of Figure 2. The authors thank the reviewers for their detailed comments. This work was supported by grants from Alberta Innovates – Technology Futures, the National Science and Engineering Research Council of Canada, and the Alberta Innovates Centre for Machine Learning.

References

Becker, J. D. (1973). A model for the encoding of experiential information. In: *Computer Models of Thought and Language*, R. C. Shank and K. M. Colby (eds), pp. 396–434. W. H. Freeman and Company, San Francisco.

Box, G. E., Jenkins, G. M., Reinsel, G. C. (2011). *Time series analysis: Forecasting and control*. Wiley.

Boots, B., Siddiqi, S. M., Gordon, G. J. (2011). Closing the learning-planning loop with predictive state representations. *International Journal of Robotics Research 30*(7):954–966.

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

Butz, M., Sigaud, O., Gérard, P., Eds. (2003). *Anticipatory Behaviour in Adaptive Learning Systems: Foundations, Theories, and Systems*, LNAI 2684, Springer.

Camacho, E. F., Bordons, C. (2004). *Model Predictive Control*. Springer.

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

Clark, A. (2013). Whatever next? Predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences 36*(3):181–204.

Cunningham, M. (1972). *Intelligence: Its Organization and Development* Academic Press, New York.

Dayan, P., Hinton, G. (1993). Feudal reinforcement learning. *Advances in Neural Information Processing Systems 5*:271–278.

Degris, T., Pilarski, P. M., Sutton, R. S. (2012). Model-free reinforcement learning with continuous action in practice. In: *Proceedings of the American Control Conference*, pp. 2177–2182.

Drescher, G. L. (1991). Made-up minds: A constructivist approach to artificial intelligence. MIT Press.

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 27*:377–442.

Hawkins, J., 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: *Proceedings of International Joint Conference on Artificial Intelligence*, pp. 1094–1099.

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

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

Littman, M. L., Sutton, R. S., Singh, S. (2002). Predictive representations of state. *Advances in neural information processing systems 14*:1555-1561.

Ljung, L. (1998). *System Identification: Theory for the User*. Prentice Hall.

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

Maei, H., Sutton, R. S. (2010). GQ(λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces. In: *Proceedings of the Third Conference on Artificial General Intelligence*, pp. 91–96.

Markoff, J. (2010). Google cars drive themselves, in traffic. *The New York Times*, October 10.

Modayil, J., White, A., Sutton, R. S. (2012) Multi-timescale Nexting in a Reinforcement Learning Robot In: *From Animals to Animats 12: 12th International Conference on Simulation of Adaptive Behavior, SAB 2012*, T. Ziemke, C. Balkenius, J. Hallam (Eds.), LNAI 7426, pp. 299–309. Springer, Heidelberg.

Oates, T., Schmill, M. D., Cohen, P. R. (2000) A method for clustering the experiences of a mobile robot that accords with human judgments. In: *Proceedings of the Seventeenth Conference of the Association for the Advancement of Artificial Intelligence*, pp. 846–851.

Oudeyer, P.-Y., Kaplan, F., Hafner, V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation 11*(2):265–286.

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.

Peters, J., Schaal, S. 2008. Natural actor-critic. *Neurocomputing 71*(7):1180–1190.

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

Pierce, D., Kuipers, B. J. (1997). Map learning with uninterpreted sensors and effectors. *Artificial Intelligence 92*(1):169–227.

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. In: *Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence*, pp. 202–207.

Singh, S., James, M. R., Rudary, M. R. (2004). Predictive state representations: A new theory for modeling dynamical systems. In: *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pp. 512–519.

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

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In: *Proceedings of the Seventh International Conference on Machine Learning*, pp. 216–224.

Sutton, R. S. (1995). TD models: Modeling the world at a mixture of time scales. In: *Proceedings of the International Conference on Machine Learning*, pp. 531–539.

Sutton, R. S. (2009). The grand challenge of predictive empirical abstract knowledge. In: *Working Notes of the IJCAI-09 Workshop on Grand Challenges for Reasoning from Experiences.*

Sutton, R. S. (2012). Beyond reward: The problem of knowledge and data. In: *Proceedings of the 21st International Conference on Inductive Logic Programming*, S. H. Muggleton, A. Tamaddoni-Nezhad, F. A. Lisi (Eds.), LNAI 7207, pp. 2–6. Springer, Heidelberg.

Sutton, R. S., Barto, A. G. (1990). Time-derivative models of Pavlovian reinforcement. In: *Learning and Computational Neuroscience: Foundations of Adaptive Networks*, pp. 497–537. MIT Press.

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

Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, Cs., Wiewiora, E. (2009). Fast gradient-descent methods for temporal-difference learning with linear function approximation. In: *Proceedings of the 26th International Conference on Machine Learning*, pp. 993–1000.

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., 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*, pp. 761–768.

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

Sutton, R. S., Szepesvári, Cs., Maei, H. R. (2009). A convergent $O(n)$ algorithm for off-policy temporal-difference learning with linear function approximation. *Advances in Neural Information Processing Systems 21*. MIT Press.

Sutton, R. S., Tanner, B. (2005). Temporal-difference networks. *Advances in neural information processing systems 17*:1377-1384.

Tedrake, R., Zhang, T., Seung, H. (2005). Stochastic policy gradient reinforcement learning on a simple 3D biped. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 2849–2854.

Thrun, S., Montemerlo, M., et al. (2006). Stanley: The robot that won the DARPA grand challenge. *Journal of Field Robotics 23*(9):661–692.

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

Wang, C. C., Thorpe, C., Thrun, S. (2003). Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 842–849.

Welch, G., Bishop, G. (1995). *An Introduction to the Kalman Filter*. Technical Report 95-041, University of North Carolina at Chapel Hill, Computer Science Dept.

White, A., Modayil, J., Sutton, R. S. (2012). Scaling lifelong off-policy learning. In: *Proceedings of the Second Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics*.

Wolpert, D., Ghahramani, Z., Jordan, M. (1995). An internal model for sensori-motor integration. *Science 269*(5232):1880–1882.

Yamashita, Y., Tani, J. (2008). Emergence of functional hierarchy in a multiple timescale neural network model: A humanoid robot experiment. *PLOS: Computational Biology 4*(11):1–18.