# Scaling Life-long Off-policy Learning

Adam White, Joseph Modayil, and Richard S. Sutton

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

June 28, 2012

**Abstract**

We pursue a life-long learning approach to artificial intelligence that makes extensive use of reinforcement learning algorithms. We build on our prior work with general value functions (GVFs) and the Horde architecture. GVFs have been shown able to represent a wide variety of facts about the world's dynamics that may be useful to a long-lived agent (Sutton et al. 2011). We have also previously shown scaling—that thousands of on-policy GVFs can be learned accurately in real-time on a mobile robot (Modayil, White & Sutton 2011). That work was limited in that it learned about only one policy at a time, whereas the greatest potential benefits of life-long learning come from learning about many policies in parallel, as we explore in this paper. Many new challenges arise in this off-policy learning setting. To deal with convergence and efficiency challenges, we utilize the recently introduced GTD($\lambda$) algorithm. We show that GTD($\lambda$) with tile coding can simultaneously learn hundreds of predictions for five simple target policies while following a single random behavior policy, assessing accuracy with interspersed on-policy tests. To escape the need for the tests, which preclude further scaling, we introduce and empirically validate two online estimators of the off-policy objective (MSPBE). Finally, we use the more efficient of the two estimators to demonstrate off-policy learning at scale—the learning of value functions for one thousand policies in real time on a physical robot. This ability constitutes a significant step towards scaling life-long off-policy learning.

## 1 Introduction

Life-long learning is an approach to artificial intelligence based on learning from a long-stream of sensorimotor interaction generated by an agent interacting with its environment, Life-long learning emphasizes continual learning by an autonomous agent over long periods of time, perhaps months or years. This big data problem requires algorithms that scale efficiently to learn a multitude of diverse facts about the large stream of sensorimotor data. We purse a novel approach to knowledge acquisition and verification based on algorithms from reinforcement learning and a life-time of sensorimotor interaction of a mobile robot.

General value functions (GVFs) provide an expressive language for representing sensorimotor knowledge about a long-lived agent's interaction with the world (Sutton et al. 2011). Knowledge is represented as approximate value functions with a reward function, an outcome function and a pseudo-termination function conditioned on a policy. GVFs provide a semantics for experiential knowledge that is grounded in the sensorimotor data and verifiable by the agent, without human intervention—essential for scalability.

In recent computational studies (Modayil, White & Sutton 2011) have shown that predictions represented as GVFs can be learned at a massive scale with a high degree of accuracy. Under on-policy sampling, a mobile robot learned thousands of predictions about future sensor readings and state variables at several time scales operating at a 100 ms time step. These predictions were shown to be accurate compared to the optimal off-line solution.

Previous work focused on a limited form of prediction, learning the consequences of a single policy. The greatest potential benefit of life-long learning comes from learning about many policies in parallel, using off-policy learning. Parallel learning introduces a new dimension of scaling not considered in typical sequential life-long learning systems and has yet to be demonstrated at scale and in real time on a robot.

Many new challenges arise in this off-policy learning setting. Scaling life-long learning in this way requires efficient learning methods that are robust under off-policy sampling. We use recently developed gradient temporal-difference learning method, GTD($\lambda$), with linear function approximation, to learn GVFs on a robot. Gradient TD methods are the only learning methods that scale linearly in the size of the feature set, require constant computation time per step, do not require memory or a forgetting process, and are therefore the only methods available for off-policy life-long learning at scale on a mobile robot. Arguably, the computational constraints of this big-data problem make batch and least-squares approaches inappropriate for life-long learning, while recent work in machine learning has highlighted the value of simple online learning methods for big data problems (see Hsu et al., 2011).

Evaluating off-policy learning at scale poses an additional challenge: determining prediction accuracy for policies that are never executed by the robot. We first show that our robot can learn hundreds of predictions about several policies, with interspersed on-policy tests. However, these tests require interrupting learning, placing an upper bound on the number of policies the robot can learn about. We propose two efficiently computable, online measures of off-policy learning progress based on the off-policy objection function (MSPBE). Using these online measures, we demonstrate learning a thousand GVFs about a thousand unique target policies in real time on a robot. Our results represent a significant step towards scaling life-long learning.

## 2 On-policy and off-policy prediction with value functions

To begin, we consider how the problem of prediction is conventionally formulated in reinforcement learning. The interaction between an agent and its environment is modelled as a discrete-time dynamical system with function approximation. On each discrete time step $t$, the agent observes a feature vector $\phi_t \in \Phi = \mathbb{R}^n$, that partially characterizes the current state of the environment. Note that the agent has no access to the underlying environment state space $S$, or to the current state $s_t \in \mathcal{S}$; the observed feature vector, $\phi_t$, is computed from information available to the agent and thus is only implicitly a function of the environmental state, $\phi_t = \phi(s_t)$. At each time step, the agent takes an action $a_t \in \mathcal{A}$, and the environment transitions into a new state producing a new feature vector, $\phi_{t+1}$.

In conventional reinforcement learning, we seek to predict at each time the total future discounted reward, where *reward* $r_t \in \mathbb{R}$ is a special signal received from the environment. More formally, we seek to learn a value function $V : S \to \mathbb{R}$, conditional on the agent following a particular policy. The time scale of the prediction is controlled by a discount factor $\gamma \in [0, 1)$. With these terms defined, the precise quantity being predicted, called the *return* $g_t \in \mathbb{R}$, is

$$g_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1},$$

and the value function is the expected value of the return,

$$V(s) = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s \right],$$

where the expectation is conditional on the actions (after $t$) being selected according to a particular policy $\pi : \Phi \times \mathcal{A} \to [0, 1]$. As is common in reinforcement learning, we estimate $V$ with a linear approximation, $V_\theta(s) = \theta^\top \phi(s) \approx V(s)$, where $\theta \in \mathbb{R}^n$.

In the most common *on-policy* setting, the policy that conditions the value function, $\pi$, is also the policy used to select actions and generate the training data. In general, however, these two policies may be different. The policy that conditions the value function is called the *target* policy because it is the target of the learning process, and in this paper we will uniformly denote it as $\pi$. The policy that generates actions and behaviour is called the *behaviour* policy, and in this paper we will denote it as $b : \Phi \times \mathcal{A} \to [0, 1]$. The most common setting, in which the two policies are the same, is called *on-policy learning*, and the setting in which they are different is called *off-policy learning*.

Conventional algorithms such as TD($\lambda$) and Q-learning can be applied with function approximation in an on-policy setting, but may become unstable in an off-policy setting. Fewer algorithms work reliably in the off-policy setting. One such algorithm is GTD($\lambda$), a gradient-TD algorithm designed to learn from off-policy sampling with function approximation (Maei, 2011). GTD($\lambda$) is an incremental prediction algorithm, similar to TD($\lambda$) (Sutton, 1988), except with an additional secondary set of learned weights $w$, and an additional step size parameter $\alpha_w$. The algorithm retains the computational advantages of TD($\lambda$): its computational complexity is $\mathcal{O}(n)$ per step, and it can operate online and in real time. Unlike TD($\lambda$), GTD($\lambda$) is guaranteed to converge under off-policy sampling and with function approximation (linear and non-linear). The following pseudocode specifies the GTD($\lambda$) algorithm.

```
Initialize $w_0$ and $e_0$ to zero and $\theta_0$ arbitrarily.
**for** each time step $t$, given observed sample $\phi_t, a_t, \phi_{t+1}$, and $r_{t+1}$ **do**
    $\delta_t \leftarrow r_{t+1} + \gamma\theta_t^\top \phi_{t+1} - \theta_t^\top \phi_t$
    $\rho_t \leftarrow \frac{\pi(a_t|\phi_t)}{b(a_t|\phi_t)}$
    $e_t \leftarrow \rho_t(\phi_t + \gamma\lambda e_{t-1})$
    $\theta_{t+1} \leftarrow \theta_t + \alpha_v(\delta_t e_t - \gamma(1-\lambda)(e_t^\top w_t)\phi_{t+1})$
    $w_{t+1} \leftarrow w_t + \alpha_w(\delta_t e_t - (\phi_t^\top w_t)\phi_t)$
**end for**
```

The GTD($\lambda$), algorithm minimizes the $\lambda$-weighted mean-square projected Bellman error

$$\text{MSPBE}(\theta, \Phi) = ||V_\theta - \Pi_\Phi T_\pi^{\lambda,\gamma} V_\theta||_D^2 \tag{1}$$

where $\Phi$ is the matrix of all possible feature vectors $\phi$, $\Pi_\Phi$ is a projection matrix that projects the value function onto the space representable by $\Phi$, $T_\pi^{\lambda,\gamma}$ is the $\lambda$-weighted Bellman operator for the target policy $\pi$ and discount factor $\gamma$, and $D$ is a diagonal matrix whose diagonal entries correspond to the state visitation frequency under the behaviour policy $b$.

## 3 An architecture for large-scale, real-time off-policy learning on robots

In addition to learning about multiple policies, our approach is to learn multiple things about each policy. Both of these cases can be captured with the notion of general value functions. We envision an architecture in which many predictive questions are posed and answered in a generalized form of value function. Each such function, denoted $v^i : \mathcal{S} \to \mathbb{R}$, predicts the expected discounted sum of the future readings of some sensor. The $i$th value function pertains to the sensor readings $r_t^{(i)}$, the policy $\pi^{(i)}$, and the time scale $\gamma^{(i)}$:

$$v^i(s) = \mathbb{E}_{\pi^{(i)}}\left[\sum_{k=0}^{\infty}(\gamma^{(i)})^k r_{t+k+1}^{(i)} \middle| s_t = s\right].$$

Off-policy methods can be used to learn approximate answers $v_\theta^{(i)}$ to predictive questions in the form of such value functions. Questions about what will happen to robot *if* it follows a behaviour different from its current behaviour, for

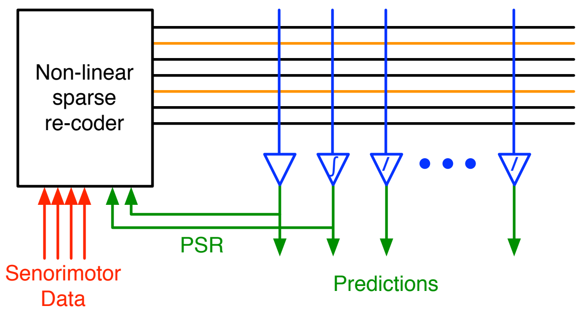

Figure 1: The Horde architecture for large-scale off-policy learning. Horde consists of a large set of independent instances of the GTD($\lambda$) algorithm (specified by triangles), updating and making predictions in parallel from a shared set of features. The features are typically a sparse encoding of the raw sensorimotor information and the predictions from the previous time step. The whole system can be operated in parallel and in real time.

example, 'what would be the effect on the rotational velocity, if my future actions consisted of clockwise rotation'. Policy-contingent questions substantially broaden the knowledge that can be acquired by the system and dramatically increases the scale of learning—millions of distinct predictive questions can be easily constructed from the space of policies, sensors and time scales.

Figure 1 provides a graphical depiction of this parallel learning architecture. This architecture, called *Horde* by Sutton et al. (2011), has several desirable characteristics. Horde can run in real time, due to the linear computational complexity of GTD($\lambda$). The architecture is potentially scalable because of the distributed nature of off-policy learning, but no experiments were performed to evaluate its ability to learn at scale in practice. The system is modular: the question specification, behaviour policy and the function approximation architecture are completely independent. As depicted in Figure 1, the predictions can be used as input to the function approximator. This enables the use of predictive state information (Littman et al., 2002) and learning of compositional predictions, similar to a TD network (Sutton and Tanner, 2005).

## 4 Large-scale off-policy prediction on a robot

The first question we consider is whether the Horde architecture supports large-scale off-policy prediction in real time on a physical robot. All our evaluations were performed on a custom-built holonomic mobile robot (see Figure 2). The robot has a diverse set of 53 sensors for detecting external entities (ambient light, heat, infrared light, magnetic fields, and infrared reflectance) and also its internal status (battery voltages, acceleration, rotational velocity, motor velocities, motor currents, motor temperatures, and motor voltages). The robot can dock autonomously with its charging station and can run continually for twelve hours without recharging.

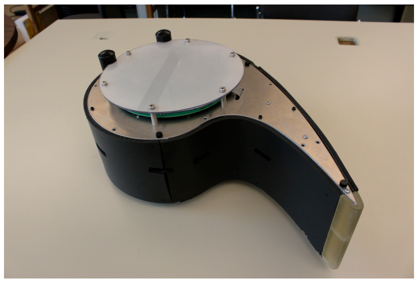

Figure 2: The mobile robot can drive into walls without harm and operate for hours without recharging.

The raw sensorimotor vector was transformed into features, $\phi_t$, by tile coding. This produced a binary vector, $\phi_t \in \{0,1\}^n$, with a constant number of 1 features. The tile coder was comprised of many overlapping tilings of single sensors and pairs of sensors. The tile coding scheme produced a sparse feature vector with $k = 6065$ components with 457 features that were ones, including one bias feature whose value was always 1. More details of the feature representation are given in previous work (Modayil, White & Sutton 2011).

Conducting a fair evaluation presents a challenge for off-policy predictions on a robot; the most direct way to evaluate a prediction about a policy is to follow that policy for a period of time and measure the return. This direct on-policy test excursion is interspersed into a baseline off-policy behaviour used for learning. We followed this procedure to evaluate the predictions learned off-policy on the robot, with learning updates being suspended during the test excursions.

To generate behaviour data, the robot was confined to a small two meter square pen, executing one of five actions: $\mathcal{A} = \{$forward, reverse, rotate clockwise, rotate counter clockwise, stop$\}$. A new action was selected for execution every 100ms. For the baseline learning behaviour, at each time step a random action was selected with a probability of 0.5, otherwise the last executed action was repeated. Normal execution was interrupted probabilistically to run a test excursion; on average an interruption occurred every 5 seconds. A test excursion consisted of selecting one of five constant action policies and following it for five seconds. After a test excursion was completed, the robot spent 2 seconds moving to the centre of the pen and then continued its random behaviour policy. The robot ran for 7.3 hours, visiting all portions of the pen many times. This produced 261,681 samples with half of the time spent on test excursions.

We used Horde to learn answers to 795 predictive questions from the experience generated by the behaviour described above. Each question $v^i$, was formed by combining a $\gamma^{(i)} \in \{0.0, 0.5, 0.8\}$, a constant action policy $\pi^{(i)}$ from $\{\pi(\cdot, forward) = 1, \pi(\cdot, reverse) = 1, \ldots, \pi(\cdot, stop) = 1\}$, and a prediction target $r^{(i)}$ from one of the 53 sensors. Each question was of the form: *at the current time t, what will be the expected discounted sum of the future values of* $r^{(i)}$ *if the robot follows* $\pi^{(i)}$*, with a constant pseudo-termination probability of* $1 - \gamma^{(i)}$ *?*. To ease comparison of the predictions across sensors with different output ranges, the values from each sensor were scaled to the maximum and minimum values in their specifications, so that the observed sensor values were bounded between [0,1]. Each time-step resulted in updates to exactly 159 GTD($\lambda$) learners in parallel (corresponding to the policies that matched the action selected by the behaviour policy). Each question used identical learning parameters: $\alpha_\theta = 0.1/457$ (457 is the number of active features), $\alpha_w = 0.001\alpha_\theta$, and $\lambda = 0.9$.

The total computation time for a cycle under our conditions was 45ms, well within the 100ms duty cycle of the robot. The entire architecture was run on a 2.4GHz dual-core laptop with 4GB of RAM connected to the robot by a dedicated wireless link.

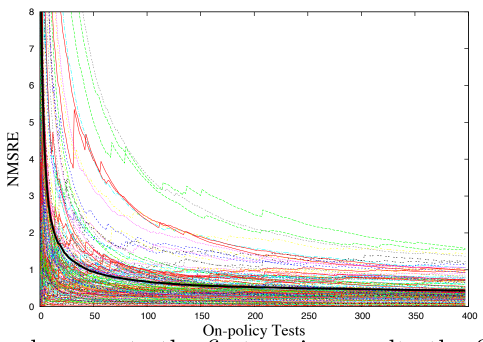

Figure 3: This graph presents the first major result: the first demonstration of learning hundreds policy-contingent predictions at 10 Hz on a consumer laptop. The x-axis is the number of relevant test excursions observed for each question. The black heavy stroke line shows the average error over the entire set of questions, and the return error has the typical exponential learning profile. The return errors are normalized by the variance in the return for each question, yielding the percentage of variance unexplained. Several individual curves exhibit non-monotonic shape due to discrepancies between the samples observed under the test excursions and during learning—difficult to avoid while evaluating performance online with individual test excursions.

With the architecture in place to update many off-policy predictions in real time on a robot, we the evaluated on-policy test performance. More precisely, on each test execution, for each of the 159 questions pertaining to the selected test policy, we compared the prediction at the beginning of the test, $\hat{v}^i(\phi_t)$, with the truncated sample return gathered during the test excursion:

$$\hat{g}^i_t = \sum_{k=0}^{50} (\gamma^{(i)})^k r^{(i)}_{t+k+1}.$$

Finally, each prediction error was normalized by the sample variance of each $\hat{g}_t^i$ over all starting configurations observed (computed off-line), yielding a normalized mean squared return error (NMSRE)[1] :

$$\text{NMSRE}_t^i = \overline{(\hat{v}^i - \hat{g}^i)^2}/\text{Var}[\hat{g}^i]. \tag{2}$$

The NMSRE represents the percentage of the variance in the returns that remains unexplained by the predictor. For the questions whose sample returns are constant and thus have a zero sample variance, we define the NMSRE to be one.

Figure 3 illustrates our main result: accurate off-policy predictions can be learned, in real time, on a physical robot at scale. These predictions were learned from a randomized behaviour policy with a shared feature representation using identical parallel instances of GTD($\lambda$). Note that no question-specific tuning of learning parameters or features was needed. Another significant result is that no divergence was observed for any question. Note the average of the NMSRE for all the questions finished below 1.0: a substantial portion of the variance in the returns is being explained by the predictions. The learning parameters were important—divergence was observed on earlier runs with more aggressive settings (both larger values of $\alpha_\theta$ and smaller values of $\alpha_w$), thus demonstrating that convergence in this off-policy setting is not trivial.

## 5 An online measure of off-policy learning progress

The accuracy of the predictions learned in the previous experiment was evaluated with the return error observed during on-policy test excursions. These tests consume considerable wall-clock time, because for each sample the system must follow the target policy long enough to capture most of the probability mass of the infinite sample return and multiple samples are required to estimate the NMSRE. Interspersing on-policy tests to evaluate learning progress places a low limit on both the number of target policies and on the time-scale given by $\gamma$.

There are other subtle deficiencies with on-policy tests. The experimenter must choose a testing regime and frequency. Depending on how often tests are executed, there is a trade-off for how often the NMSRE is updated. Changes in the environment and novel robot experiences can cause inaccurate NMSRE estimates if the majority of time-steps are used for training. Testing with greater frequency ensures the estimated NMSRE closely matches current prediction quality, but slows learning.

In the function approximation setting, we propose instead to estimate the MSPBE. The GTD($\lambda$) algorithm does not minimize the NMSRE, which mea-

[1] We use $\overline{x}$ to denote the exponential trace (also called the exponentially weighted average) of samples of $x_t$; this is computed for a fixed time constant $\tau$ by $\overline{x} \equiv \text{trace}(x,t) = \frac{1}{\tau}x_t + (1 - \frac{1}{\tau})\text{trace}(x, t-1)$.

sures prediction accuracy relative to sample returns, ignoring function approximation error. For an arbitrary question the NMSRE will never go to zero, though it does provide an indication of the quality of the feature representation. The GTD($\lambda$) algorithm instead minimizes the MSPBE. Under some common technical assumptions, the MSPBE will converge to a zero error. The MSPBE can be estimated in real time during learning, and it provides an up-to-date measure of performance without sacrificing valuable robot data for evaluation.

Using the derivation given by Sutton et al. (2009), we can rewrite this error in terms of expectations:

$$\begin{aligned}
\text{MSPBE}(\theta) &= ||v_\theta - \Pi T v_\theta||_B^2 && (3)\\
&= ||\Pi(v_\theta - T v_\theta)||_B^2 && (4)\\
&= (\Pi(v_\theta - T v_\theta))^\top B(\Pi(v_\theta - T v_\theta)) && (5)\\
&= (v_\theta - T v_\theta)^\top \Pi^\top B \Pi (v_\theta - T v_\theta) && (6)\\
&= (v_\theta - T v_\theta)^\top B^\top \Phi(\Phi^\top B \Phi)^{-1}\Phi^\top B(v_\theta - T v_\theta) && (7)\\
&= (\Phi^\top B(T v_\theta - v_\theta))^\top (\Phi^\top B \Phi)^{-1}\Phi^\top B(T v_\theta - v_\theta) && (8)\\
&= \mathbb{E}_b[\delta\phi]^\top \mathbb{E}_b[\phi\phi^\top]^{-1}\mathbb{E}_b[\delta\phi] && (9)
\end{aligned}$$

The GTD($\lambda$) algorithm uses a second set of modifiable weights, $w$, to form a quasi-stationary estimate of the last two terms, namely the product of the inverse feature covariance matrix with the expected TD-update. This leads to the following linear-complexity approximation of the MSPBE:

$$\text{MSPBE}(\theta) \approx (\mathbb{E}_b[\delta\phi])^\top w. \qquad (10)$$

The expected TD-update term, $\mathbb{E}_b[\delta\phi]$, can be approximated with samples of $\delta_t e_t$ (switching from the forward view to backward view), where $e_t$ is the eligibility trace vector. Additionally, the prediction error can be non-zero on samples where the target policy does not agree with the behaviour policy, $\pi^{(i)}(\phi_t, a) \neq b(\phi_t, a)$. The importance sampling ratio, $\frac{\pi^{(i)}(\phi,a)}{b(\phi,a)}$, can be used to account for these effects. This leads to two natural incremental algorithms for sampling the current MSPBE:

$$\text{MSPBE}_{t,vector} = \overline{\delta e}^\top w_t, \qquad (11)$$

and

$$\text{MSPBE}_{t,scalar} = \overline{\delta e^\top w}. \qquad (12)$$

Here, the exponential traces for both $\text{MSPBE}_{t,vector}$ and $\text{MSPBE}_{t,scalar}$ are updated on each time step proportionally to $\frac{\pi^{(i)}(\phi,a)}{b(\phi,a)}$. The first measure is a more accurate approximation of Equation 10, but the second requires only storing a single real-valued scalar.

The first step in evaluating our online estimates of the MSPBE, is to compare them with the exact values of the MSPBE on a simulation domain. We used a simple 7 state Markov chain with an absorbing state on each end, deterministic

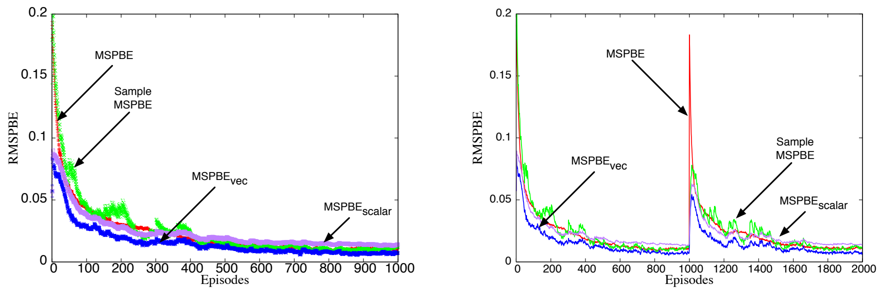

Figure 4: Comparison of the online estimates of the MSPBE and the true and sample MSPBE on a simple Markov chain. The left figure figure shows that the measures provide a good estimate of MSPBE matching the profile and magnitude on this simple domain. The right figure compares how the online estimates react to a major change during learning. After 1000 episodes the primary weight vector $\theta$ was set to random values in $[0, 1]$. Again, the online estimates track the true and sample MSPBE closely.

transitions and episodes beginning in the middle of the chain. Transitioning into the right-side terminal state produced a reward of 1.0, all other transitions incurred 0 reward. We used the inverted feature representation of Sutton et al (2009).

To determine the validity of our new measures we compared the vector and scalar MSPBE estimates with the true MSPBE (Equation 4) and the expensive sample estimate of the MSPBE (Equation 9). The computation of the true MSPBE requires complete knowledge of the chain MDP. The sample MSPBE requires an incremental estimate of the expected feature covariance matrix and a inverse operation. The sample MSPBE represents the best possible sample-based estimate of the MSPBE; our online measures can not be expected to track the true MSPBE better than the sample MSPBE. In this experiment we used a single instance of GTD with $\alpha_\theta = 0.05$, $\alpha_w = 0.1$ and $\lambda = 0.0$ and results were averaged over 100 independent runs. The target policy selected the move-right action with probability 0.95 and a behaviour policy that selected move-right with probability 0.2.

Figure 4 (left) illustrates the results of the chain experiment comparison: both our online measures provide an accurate estimate of the MSPBE on a simple chain domain. Figure 4 (right) compares the effect of a change in the world during learning. After 1000 episodes the $\theta$ weights were set to random values in $[0, 1]$. The secondary weights $w$, and traces used in computing the online estimates were not reset. The results depicted in Figure 4 (right) illustrate that the online estimates, sample MSPBE and the true MSPBE all react similarly to the change.

To evaluate the online MSPBE estimates on the robot, we compare aggregate error curves, averaged over all questions, on tasks where the robot experiences a significant change, similarly to the chain experiment. The robot was run exactly as before, with a subset of the predictions learned ($\gamma = 0.8$), for six hours. This time, the learned weight vector of each question $\theta^{(i)}$, was set to zero after 40000 time steps. This change effectively reinitializes each question and effects the accuracy of all the predictions. In this experiment, we recorded the NMSRE, $\text{MSPBE}_{t,vector}$ and $\text{MSPBE}_{t,scalar}$ on every time step for 265 questions, except during test excursions. Note that the NMSRE is only updated after a test completes, while the MSPBE measures are updated on every non-test time-step.

Figure 5 compares the convergence profile and reaction to change of the three error measures in terms of training time. As in the chain experiments, all three measures react quickly to the change. Note that both MSPBE estimates are initially at zero, as the vector $w$ takes time to adapt to a useful value. Finally, note that the $\text{MSPBE}_{t,vector}$ and $\text{MSPBE}_{t,scalar}$ exhibit very similar trends, indicating that the Bellman error can be estimated with minimal storage requirements.

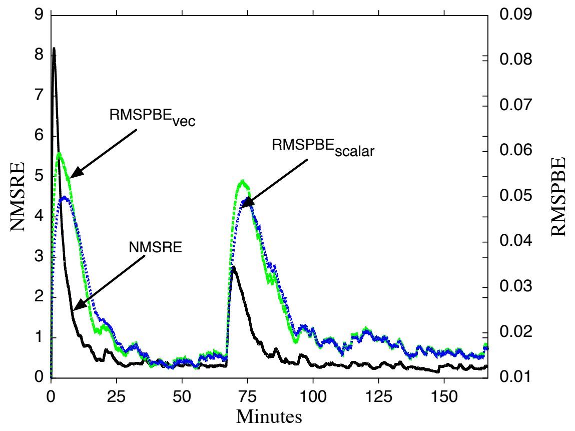

Figure 5: This figure compares the NMSRE, with two estimates of the MSPBE, averaging performance over 265 predictive questions. This figure illustrates several important points that validate the MSPBE as a useful error measure. The MSPBE measures as computed off-policy have the same shape as the NMSRE which requires an on-policy excursion in the behaviour. The MSPBE estimates converge more slowly than the NMSRE indicating that although test performance is not improving, learning is still occurring. The MSPBE estimates react more slowly to the change because the secondary weights $w$ where not also reset; their incorrect values affect the estimates of the MSPBE long after the change. The MSPBE measures react quickly to a change in the environment. This experiment shows that the scalar estimate of the MSPBE performs almost identically to the vector version. This graph illustrates a significant result: off-policy learning progress can be efficiently estimated online, without time consuming on-policy tests. Additionally that the NMSRE requires twice as twice as much wall-clock time as the MSPBE estimates.

## 6 Large-scale off-policy prediction, with many target policies

Free from the limitations of physically performing test excursions to evaluate predictions, we can learn about a much larger set of questions. In this section, we consider scaling the number of target policies and prediction time scales (magnitude of $\gamma$).

To increase the space of target policies, and still maintain a small set of finite actions, we consider discrete-action linearly parametrized Gibbs policy distributions:

$$\pi_u(a) = \frac{\exp(-u^\top \Psi_a)}{\sum_{a' \in \mathcal{A}} \exp(-u^\top \Psi_{a'})}$$

where $u$ is a vector of policy parameters. The feature vector for each action, $\Psi_a \in \mathbb{R}^{n|\mathcal{A}|}$, has a copy of $\phi_t$ as a subvector in an otherwise zero vector; and for each action the copy is offset by $n$ so that $a \neq a' \implies \Psi_a^\top \Psi_{a'} = 0$. Random policies are generated by selecting 60 components of $u$ at random and assigning each a value independently drawn from the uniform distribution over $[0, 1]$.

In this final experiment, we tested how well our architecture scales in the number of target policies. The robot's behaviour was the same as before, but now learning was enabled on every step of the 7 hours experience. The questions were formed by sampling $\gamma$ values from $\{0.0, 0.5, 0.8, 0.95\}$, reward from the full set of sensors with 1000 randomly generated policies. The value of $\gamma = 0.95$ corresponds to a 2 second prediction and would require over 30 seconds to accurately evaluate using the NMSRE. The 1000 questions, evaluated according to MSPBE$_{t,scalar}$, were learned with a cycle time of 85ms on a 4-core desktop computer with 16 G of Ram; satisfying our real-time requirement of 100ms.

Figure 6 presents the results of this experiment, namely that learning the temporally-extended consequences of many (untestibly many) different behaviours is possible in real time. The learning progress is measured by the MSPBE, which by the results in the previous section will be strongly coupled to on-policy prediction errors. Note that the ability to monitor learning progress across so many different behaviours is only possible due to the availability of the MSPBE. By acquiring many predictions about many different courses of behaviour, the robot can acquire detailed partial models of the dynamics of its environmental interaction.

## 7 Related Work

Many of the ideas in this paper have precursors in the literature. The idea of policy-contingent predictions was developed along with the options framework for temporal abstraction (Sutton, Precup & Singh, 1999). Learning off-policy under function approximation was developed by importance sampling (Precup et al., 2006) in an approach that runs online, but can exhibit exponentially slow learning progress. Learning about many different policies can also support active

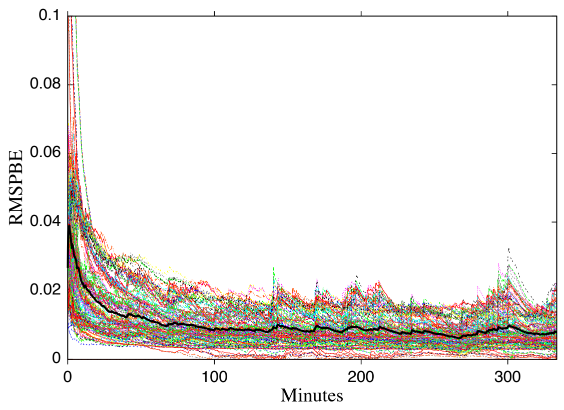

Figure 6: Scaling off-policy learning to 1000 robot policies. The estimated mean squared projected Bellman error for off-policy predictions for 1000 distinct randomly generated policies. The heavy stroke black line denotes the average estimated error over the full set of questions. The curves provide a clear indication of learning progress for each prediction. This result provides a clear demonstration of the significance of estimating the MSPBE incrementally during learning. Massively scaling the number of policies on which to condition predictions is only possible with an online performance measure.

exploration, an idea that has been explored in related work on curiosity-based learning (Singh et al., 2005).

The idea of building models from data has been explored, but not in the off-policy real-time setting. Thrun and Mitchell (1995) showed learning of sensor models offline. Many Bayesian approaches for online state estimation in robotics (e.g., Thrun et al., 2005) process a vast volume of observations in real time, but they do not learn system dynamics online. A Kalman filter (Kalman, 1965) can be viewed as learning an adaptive model of dynamics online, but it is only appropriate for small models with well-understood dynamics. Atkeson and Schaal (1997) showed learning of small one-time-step models. Kober and Peters (2011) showed on-policy episodic learning on a robot.

A recent online spectral approach (Boots, Siddiqi & Gordon, 2011) finds small predictive state models for robotics. They use a sophisticated incremental method for making multiple action-conditional predictions, but it is unclear if this approach can operate under real-time constraints. The previous work that introduced the Horde architecture (Sutton et al., 2011) also demonstrated parallel off-policy learning on a robot. Their work was suggestive but did not demonstrate that the approach scaled in practice. Each experiment required a unique parameter set, function approximation scheme, and behaviour policy; this overhead is not practical for learning thousands of predictions. This paper shows that a large set of diverse, policy-contingent predictions can be learned using a shared feature set, common learning parameters, and data generated by a single random behaviour policy.

## 8 Conclusions and future work

We provided the first demonstrations of large scale off-policy learning on a robot. We have shown that gradient TD methods can be used to learn hundreds of temporally-extended policy-contingent predictions from off-policy sampling. To achieve this goal required resolution of several challenges unique to the off-policy setting. Most significantly we have developed on online estimate of off-policy learning progress based on the Bellman error that does not increase the computational complexity of the horde architecture, can be sampled without interrupting learning and has good correspondence with the traditional mean squared prediction error. The addition of policy contingent, what-if, questions dramatically increases the scope and scale of questions that can be learned by horde providing further evidence of the significance of horde for life-long learning.

Our experiments, while on a robot, are limited but there are several immediate directions for future work. The questions learned here are predictive questions about a policy, general value functions can also support learning control policies using greedy-gq (Maei 2011) a control variant of GTD($\lambda$) with the same linear complexity. Predictive accuracy improvements should be achieved by employing adaptive behaviour policies (e.g., curiosity) and more powerful function approximators, while significantly increased scaling (more predictions and larger feature vectors) can be obtained with more computational resources.

## 9 References

Atkeson, C. G., Schaal, S. (1997). Robot learning from demonstration. In *Proc. 14th Int. Conf. on Machine Learning*, pp. 12–20.

Baird, L. C. (1995). Residual algorithms: Reinforcement learning with function approximation. In *Proc. 12th Int. Conf. on Machine Learning*, pp. 30–37.

Boots, B., Siddiqi, S., Gordon, G. (2011). An online spectral learning algorithm for partially observable nonlinear dynamical systems. In *Proc. Conf. of the Association for the Advancement of Artificial Intelligence.*

Hsu, D., Karampatziakis, N., Langford, J., Smola, A. J. (2011). Parallel Online Learning. In *The Computing Research Repository.*

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Trans. ASME, Journal of Basic Engineering 82*:35–45.

Kober, J., Peters, J. (2011). Policy search for motor primitives in robotics. *Machine Learning 84*:171–203.

Kolter, J. Z. (2011). The fixed points of off-policy TD. In *Advances in Neural Information Processing Systems 24.*

Littman, M. L., Sutton, R. S., Singh, S. (2002). Predictive representations of state. In *Advances in Neural Information Processing Systems 14*, pp. 1555–1561.

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

Modayil, J., White, A., Sutton, R. S. (2012). Multi-timescale Nexting in a Reinforcement Learning Robot. *Proc. 12th Int. Conf. on Adaptive Behaviour.*

Precup, D., Sutton, R. S., Paduraru, C., Koop, A., Singh, S. (2006). Off-policy learning with recognizers. In *Advances in Neural Information Processing Systems 18*.

Quadrianto, N., Smola, A., Caetano, T., Vishwanathan, S. V. N., Petterson, J. (2010). Multitask learning without label correspondences. In *Advances in Neural Information Processing Systems 23*, pp. 1957–1965.

Singh S., Barto, A. G., Chentanez, N. (2005). Intrinsically motivated reinforcement learning. In *Advances in Neural Information Processing Systems 17*, pp. 1281–1288.

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

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

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., Tanner, B. (2005). Temporal-difference networks. In *Advances in Neural Information Processing Systems 17*, pp. 1377–1384.

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 *Proc. 26th Int. Conf. on Machine Learning.*

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. *Proc. 10th Int. Conf. on Autonomous Agents and Multiagent Systems.*

Talvitie, E., Singh, S. (2011). Learning to make predictions in partially observable environments without a generative model. *Journal of Artificial Intelligence Research 42*:353–392.

Thrun, S., Burgard, W., Fox, D. (2005). *Probabilistic Robotics.* MIT Press.

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