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, Canada T6G 2E8

***Abstract*—In this paper, we pursue an approach to scaling life-long learning using parallel, off-policy reinforcement-learning algorithms. In life-long learning, a robot continually learns from a life-time of experience, slowly acquiring and applying skills and knowledge to new situations. Many of the benefits of life-long learning are a result of scaling the amount of training data, processed by the robot, to long sensorimotor streams. Another dimension of scaling can be added by allowing off-policy sampling from the unending stream of sensorimotor data generated by a long-lived robot. Recent algorithmic developments have made it possible to apply off-policy algorithms to life-long learning, in a sound way, for the first time. We assess the scalability of these off-policy algorithms on a physical robot. We show that hundreds of accurate multi-step predictions can be learned about several policies in parallel and in realtime. We present the first online measures of off-policy learning progress. Finally we demonstrate that our robot, using the new off-policy measures, can learn 8000 predictions about 300 distinct policies, a substantial increase in scale compared to previous simulated and robotic life-long learning systems.**

Life-long learning is an approach to artificial intelligence based on the idea that learning from a life-time of experience, as humans do, will make learning complex skills and abstract knowledge, on a robot, more tractable. A long-lived robot's goal is to incrementally acquire increasingly sophisticated knowledge and a growing library of re-useable skills (policies) from interacting with the environment in an open ended manner. The life-long learning setting provides a natural framework for developing and evaluating incremental representation search, self-motived learning like curiosity and autonomous task generation.

Many of the positive attributes of life-long learning can be attributed to scaling. Life-long learning dramatically increases the scale of the artificial intelligence problem in the amount of data a robot must process. As opposed to data poor problems, where each sample is costly, a long-lived agent faces an infinite stream of data with an infinite number of things to learn about. A robot can more effectively learn, reuse and combine existing knowledge if their is no short time time horizon during which specific task performance will be evaluated. Similar to recent trends in internet-scale machine learning, life-long learning requires computationally-congenial online algorithms that can process massive streams of data realtime and on a robot.

A further dimension of scaling can be added by allowing off-policy sampling. In the off-policy setting, the agent learns about many policies, in parallel, while selecting actions according to a different behaviour policy. Off-policy sampling could dramatically scale life-long learning by enabling learning of thousands or millions of predictions and policies from a single unending stream of sensorimotor data. Unfortunately, off-policy sampling is well known to cause divergence of online linear learning methods, like Q-learning (Sutton and Barto., 1998). Formal convergence guarantees are particularly important in systems that generate new goals autonomously—a natural aspiration of any life-long learning robot.

Recent developments have made it possible to apply off-policy algorithms to life-long learning, in a sound way, for the first time. Our approach is to learn thousands of predictions represented as general value functions (GVFs). GVFs provide an expressive language for representing sensorimotor knowledge about a long-lived agent's interaction with the world (Sutton et al., 2011) and have also been used to improve prosthetic control (Pilarski et al., 2012). We use a new gradient temporal-difference (TD) method, GTD(λ), with linear function approximation, to learn many value functions, in parallel on a robot. Gradient TD methods are the only algorithms that scale linearly in the number of features, require constant computation time per step, and are guaranteed to converge under off-policy sampling. They are, therefore, the only methods suitable for off-policy life-long learning at scale on a robot.

In our first set of experiments, we assess whether off-policy methods can be applied at scale and in realtime on a physical robot. We demonstrate that hundreds of multi-step predictions can be learned about five simple policies, with no divergence cases, while maintaining an update time of less than 100ms. We show that these predictions are accurate via interspersed on-policy tests, indicating that the predictions explain a significant amount of variance in the tests. This result is the first demonstration of large-scale, sound off-policy learning on a physical robot.

Our second contribution is the development of an new fully-incremental method for accessing off-policy learning progress. Interspersed on-policy tests require interrupting learning and executing a policy in order to test predictions about it. This consumes valuable learning time and prevents scaling the number of policies to be learned about. We introduce two efficiently computable, online measures of off-policy learning progress based on the off-policy objective function (MSPBE). Using these online measures, we demonstrate learning 8000 GVFs about 300 unique target policies in realtime on a robot. These results demonstrate the novel ability of an agent to measure the accuracy of its predictions directly from off-policy experience, which is an important attribute for large-scale life-long learning systems.

I. 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 $\phi_t \in \Phi \subset \mathbb{R}^n$, that partially characterizes the current state of the environment, $s_t \in \mathcal{S}$. Note that the agent has no access to the underlying environment state; the observed feature vector, ϕ_t, is computed from information available to the agent at the current time step 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, ϕ_{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, π, 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 π. The policy that generates actions is called the *behavior* 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(λ) and Q-learning can be applied with function approximation in an on-policy setting, but may become unstable in an off-policy setting (Maei, 2011). Fewer algorithms work reliably in the off-policy setting. One reliable algorithm is GTD(λ), a gradient-TD algorithm designed to learn from off-policy sampling with function approximation (Maei, 2011). GTD(λ) is an incremental learning algorithm, similar to TD(λ) (Sutton & Barto, 1998), except with an additional secondary set of learned weights w, and an additional step size parameter α_w. The algorithm retains the computational advantages of TD(λ): its computational complexity is $\mathcal{O}(n)$ per step, and thus can be used online and in realtime. Unlike TD(λ), GTD(λ) is guaranteed to converge under off-policy sampling and with function approximation (linear and non-linear). The following pseudocode specifies the GTD(λ) 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**
    $\delta_t \leftarrow r_{t+1} + \gamma\theta_t^\top \phi_{t+1} - \theta_t^\top \phi_t$ //calculate td-error
    $\rho_t \leftarrow \frac{\pi(a_t|\phi_t)}{b(a_t|\phi_t)}$ //importance sampling correction
    $e_t \leftarrow \rho_t(\phi_t + \gamma\lambda e_{t-1})$ //update traces
    //update weight vectors
    $\theta_{t+1} \leftarrow \theta_t + \alpha_\theta(\delta_t e_t - \gamma(1-\lambda)(e_t^\top w_t)\phi_{t+1})$
    $w_{t+1} \leftarrow w_t + \alpha_w(\delta_t e_t - (\phi_t^\top w_t)\phi_t)$
**end for**
```

The GTD(λ), algorithm minimizes the λ-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 Φ is the matrix of all possible feature vectors ϕ, Π_Φ is a projection matrix that projects the value function onto the space representable by Φ, $T_\pi^{\lambda,\gamma}$ is the λ-weighted Bellman operator for the target policy π and discount factor γ, and D is a diagonal matrix whose diagonal entries correspond to the state visitation frequency under the behavior policy b. A more detailed discussion of GTD and the MSPBE can be found in Maei's thesis (2011).

II. 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 are captured with the idea of general value functions. We envision a setting where 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. Said differently, γ is used to summarize future values of the prediction target, which has been shown to be a good model for multi-step sensorimotor predictions in previous work (Modayil et al., 2012). The ith 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 to predictive questions in the form of approximate value functions, $v_\theta^{(i)}$. These questions are about what will happen to a robot *if* it follows a behavior different from its current behavior, for 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.

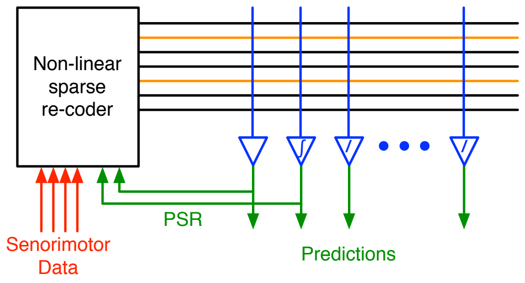

Fig. 1. The Horde architecture for large-scale off-policy learning. Horde consists of a large set of independent instances of the GTD(λ) algorithm (specified by triangles), updating and making predictions in parallel from a shared set of features. The features are typically a sparse encoding of the raw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

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 realtime, due to the linear computational complexity of GTD(λ). The system is modular: the question specification, behavior policy and the function approximation architecture are completely independent. As depicted by lines labeled 'PSR' in Figure 1, the predictions can be used as input to the function approximator. Although not explored in this work, this capability enables the use of predictive state information (Littman et al., 2002). The architecture is certainly scalable in its distributed specification of predictive questions, but no previous work has illustrated that the predictions can be learned at scale by off-policy methods.

III. 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 realtime 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.

The raw sensorimotor vector was transformed into features, ϕ_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 et al., 2012).

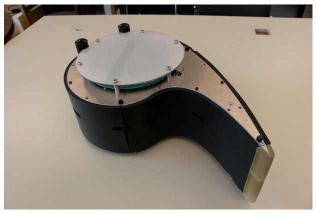

Fig. 2. The mobile robot can drive into walls without harm and operate for hours without recharging.

To generate behavior 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 behavior, 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 five seconds. A test excursion consisted of selecting one of five constant action policies and following it for five seconds with learning disabled. After a test excursion was completed, the robot spent 2 seconds moving to the centre of the pen and then resumed the random behavior policy and learning. 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 behavior 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(λ) learners in parallel (corresponding to the policies that matched the action selected by the behavior 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 $\lambda = 0.9$. Replacing traces were used and trace values below 0.01 where set to zero.

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.

With the architecture in place to update many off-policy predictions in realtime on a robot, we 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, $\hat{v}_\theta^{(i)}(\phi_t)$, with the truncated sample return gathered during

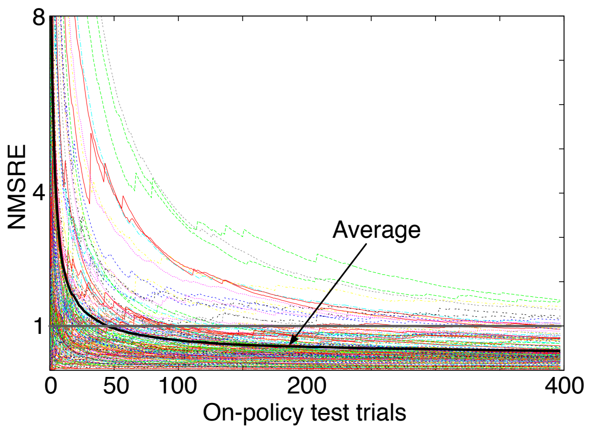

Fig. 3. This graph shows a subsampling of the predictions learned in parallel and off-policy. Hundreds of policy-contingent predictions are learned at 10 Hz on a consumer laptop. The x-axis is the number of relevant test excursions observed for each question. The black heavy stroke line shows the average 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 fraction of variance unexplained. Several individual curves exhibit non-monotonic shape due to discrepancies between the samples observed under the test excursions and during learning.

the test excursion: $g_t^{(i)} = \sum_{k=0}^{50} (\gamma^{(i)})^k r_{t+k+1}^{(i)}$. Finally, each prediction error was normalized by the sample variance of each $g_t^{(i)}$ over all starting configurations observed (computed off-line), yielding a normalized mean squared return error (NMSRE):

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

We use $\overline{x}$ to denote the exponential trace (or exponentially moving average) of samples of x_t with a time period (or decay rate) of 0.005. The NMSRE represents the fraction 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 realtime, on a physical robot at scale. These predictions were learned from a randomized behavior policy with a shared feature representation using identical parallel instances of GTD(λ). No question-specific tuning of learning parameters or features was needed and no divergence was observed for any question. 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.

IV. 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 robot 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 γ.

There are other, slightly more 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 result in 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.

We propose to use the MSPBE to measure off-policy learning progress. The GTD(λ) algorithm does not minimize the NMSRE under function approximation. NMSRE measures prediction accuracy relative to sample returns, and ignores function approximation. 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(λ) algorithm instead minimizes the MSPBE. Under some common technical assumptions, the MSPBE will converge to a zero error. The MSPBE can be estimated in realtime during learning, and it provides an up-to-date measure of performance without sacrificing valuable robot time for evaluation.

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

$$\text{MSPBE}(\theta) = ||v_\theta - \Pi T v_\theta||_B^2 = \mathbb{E}_b[\delta e]^\top \mathbb{E}_b[\phi\phi^\top]^{-1} \mathbb{E}_b[\delta e]$$

The GTD(λ) 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:

$$\text{MSPBE}(\theta) \approx (\mathbb{E}_b[\delta e])^\top w. \tag{3}$$

The expected TD-update term, $\mathbb{E}_b[\delta e]$, can be approximated with samples of $\delta_t e_t$, 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 behavior 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:

$$\text{MSPBE}_{t,vector} = \overline{\delta e}^\top w_t, \tag{4}$$

and

$$\text{MSPBE}_{t,scalar} = \overline{\delta e^\top w}. \tag{5}$$

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 time scale of the trace was set to 20 seconds (equivalent to 0.05 decay used in experiment 1). The first measure is perhaps a more accurate implementation of Equation 3, but the second requires only storing a single real-valued scalar.

As an evaluation of the online MSPBE estimates, we compare aggregate error curves, averaged over a subset of questions, on tasks where the predictions experience a sharp, significant perturbation. The idea here is to measure how quickly our online measures react to an event that causes the majority of the robot's predictions to become incorrect: similar to moving the robot to a room with different lighting conditions and surface friction. 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 prediction $\theta^{(i)}$, was set to zero after 40000 time steps. In this experiment, we recorded the NMSRE, MSPBE$_{t,vector}$ and 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 4 compares the convergence profile and reaction to change of the three error measures in terms of training time. All three measures reacted almost instantly to the change. Furthermore, the learning rate, after the change, reflected by the online MSPBE estimates was comparable with the rate exhibited by the NMSRE. 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 MSPBE$_{t,vector}$ and MSPBE$_{t,scalar}$ exhibit very similar trends, indicating that the Bellman error can be estimated with minimal storage requirements.

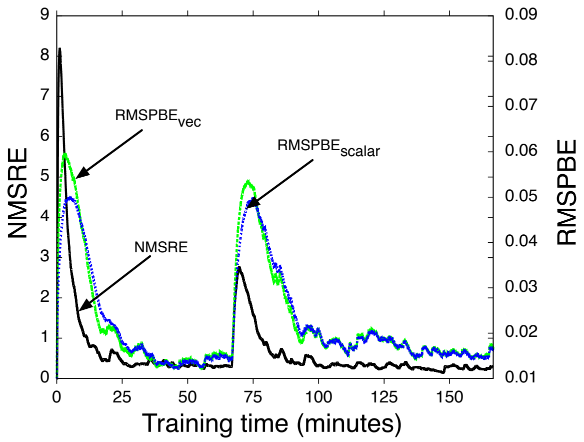

Fig. 4. This graph shows that off-policy learning progress can be efficiently estimated online, without time consuming on-policy tests. This figure compares the NMSRE, with two estimates of the MSPBE, averaging performance over 265 predictive questions. This figure illustrates several important points that validate the MSPBE as a useful error measure. The MSPBE measures as the same shape as the NMSRE that requires an on-policy excursion in the behavior. Also, the MSPBE measures react quickly to changes. However, the MSPBE estimates converge more slowly than the NMSRE indicating that although test performance is not improving, the error estimates are still improving.

V. 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 demonstrate scaling with substantial increases in the number of target policies and prediction time scales (magnitude of γ).

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 ϕ_t as a subvector in an otherwise zero vector; and for each action the copy is offset by n so that $a \neq a' \implies \Psi_a^\top \Psi_{a'} = 0$. Each policy is generated by selecting 60 components of u at random and then assigning each component 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 behavior was the same as before, but now learning was enabled on every step of the 7 hours experience. The questions were formed by sampling γ values from $\{0.0, 0.5, 0.8, 0.9, 0.95\}$, reward from the full set of sensors, and 300 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 8000 questions, evaluated according to MSPBE$_{t,scalar}$, were learned with a cycle time of 95ms on a 4-core laptop computer with 8 GB of RAM; satisfying our realtime requirement of 100ms.

Figure 5 presents the results of this experiment, namely that learning the temporally-extended consequences of many different behaviors is possible in realtime. The learning progress is measured by the RMSPBE, which, by the results in the previous section, will be strongly coupled to on-policy prediction errors. The ability to monitor learning progress across so many different behaviors is only possible due to the availability of the MSPBE.

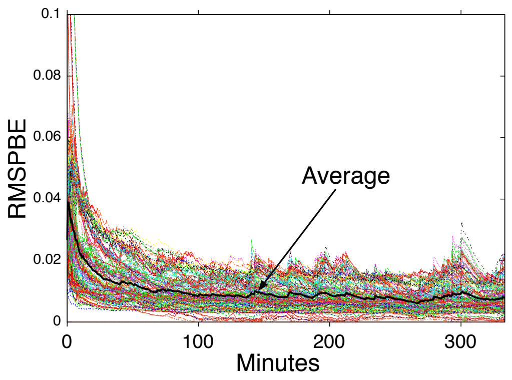

Fig. 5. Scaling off-policy learning to 8000 robot policies. The estimated mean squared projected Bellman error for a sub-sample of the predictions about 300 distinct randomly generated policies. The heavy stroke black line denotes the average estimated error over the full set of questions. The curves provide a clear indication of learning progress for each prediction. The asperous appearance of individual curves is due to randomized nature of the target policies. 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.

VI. Related Work

Although many of the ideas in this paper have precursors in the literature, no previous work has demonstrated online large-scale off-policy learning with sound convergent algorithms with function approximation on a physical robot. Thrun and Mitchell (1995) performed some of the first experiments in life-long learning on a robot, learning a few dozen predictions from on-policy experience. Singh et al. (1995) explored off-policy learning and intrinsic motivation in a tabular simulation world. Oudeyer et al. (2007) explored using curiosity to direct a robot dog to sequentially (on-policy) interact and learn about an increasingly difficult sequence of behaviours. Learning parameterized models is a common approach to learning from demonstration (Atkeson and Schaal (1997) and Kober and Peters (2011)) but has only been demonstrated on individual control tasks. Finally probabilistic robotics approaches (e.g., Thrun et al., 2005) and large-scale visual SLAM systems (Cummins & Newman, 2009) process a vast volume of observations, but do not learn online.

Outside of life-long learning most related work does not employ sound off-policy methods or is limited to small scale experiments. Prior work on options (Sutton et al., 1999) and TD-nets with options (Sutton et al., 2006) learned multi-step predictions from off-policy samples in a small simulation worlds with non-sound algorithms. A recent spectral approach to predictive state representations (Boots, Siddiqi & Gordon, 2011) is capable of learning online but is limited to on-policy sampling. Finally, Sutton et al. (2011) performed the first experiments with gradient TD methods on a robot, but was limited to a learning a maximum of 7 policies in parallel.

VII. Discussion and future work

Off-policy learning explicitly decouples the policy used to select actions on the robot and the target policies to be learned. It would be natural to adapt the behavior policy, using reinforcement learning, to maximize learning progress; creating a self-motived learning system. For example, the behavior policy could be adapted by an actor-critic agent to maximize a reward based on the sum of the prediction error of each GVF. Without convergent gradient TD methods, there was previously no way to implement and evaluate curiosity and self-motivation without restricting the system to tabular simulations (Singh et al., 2005), sequential learning (Oudeyer et al., 2008) or using methods without convergence guarantees (Schmidhuber, 1991).

Although this paper focused exclusively on off-policy prediction at scale, there are natural extensions that enable learning control policies from off-policy training. Greedy-GQ (Maei, 2011) is a control variant of GTD(λ), that can be used to learn control policies with the same linear complexity.

VIII. Conclusions

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 thousands of temporally-extended policy-contingent predictions from off-policy sampling. To achieve this goal, we addressed several challenges unique to the off-policy setting. Most significantly, we have developed the first online estimate of off-policy learning progress based on the Bellman error that 1) does not increase the computational complexity of the Horde architecture, 2) can be sampled without interrupting learning, and 3) has good correspondence with the traditional mean squared prediction error. we have shown that parallel, policy-contingent updating can substantially increases the scale of life-long learning.

Acknowledgments

The authors thank Mike Sokolsky for creating the Critterbot and Thomas Degris for assistance with the RLPark software infrastructure. 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.

IX. References

Atkeson, C. G., Schaal, S. (1997). Robot learning from demonstration. In *Proceedings of the 14th International Conference on Machine Learning*,12–20.

Boots, B., Siddiqi, S., Gordon, G. (2011). An online spectral learning algorithm for partially observable nonlinear dynamical systems. In *Proceedings of the 25th Conference of the Association for the Advancement of Artificial Intelligence*.

Cummins, M., Newman, P. (2009). Highly Scalable Appearance-Only SLAM - FAB-MAP 2.0 In *The Proceedings of the 4th Conference Robotics: Science and Systems*.

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

Littman, M. L., Sutton, R. S., Singh, S. (2002). Predictive representations of state. In *Advances in Neural Information Processing Systems 14*, 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. In *Proceedings of the 12th International Conference on Simulation of Adaptive Behavior*, LNAI 7426, 299–309

Oudeyer, P. Y., Kaplan, F., Hafner, V. (2007). Intrinsic Motivation Systems for Autonomous Mental Development. In *IEEE Transactions on Evolutionary Computation 11*, 265–286

Pilarski, P.M., Dawson, M.R., Degris, T.,Carey, J.P., Sutton, R.S. (2012). Dynamic Switching and Real-time Machine Learning for Improved Human Control of Assistive Biomedical Robots. In *Proceedings of the 4th IEEE International Conference on Biomedical Robotics and Biomechatronics*, 296–302.

Schmidhuber J. (1991). A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proceedings of the 1st International Conference on Simulation of Adaptive Behavior*, 222–227.

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

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. In *Artificial Intelligence 112*:181–211.

Sutton, R. S., Rafols, E. J., Koop, A. (2006). Temporal abstraction in temporal-difference networks. In *Advances in Neural Information Processing Systems 18*.

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*.

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 the10th International Conference on Autonomous Agents and Multiagent Systems*.

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

Thrun, S., Mitchell, T. (1995). Lifelong robot learning. *Robotics and Autonomous Systems 15*: 25–46.