Adapting the Function Approximation Architecture in Online Reinforcement Learning*

John D. Martin†
Department of Computing Science
University of Alberta / Amii
Edmonton, AB, Canada
jmartin8@ualberta.ca

Joseph Modayil†
DeepMind
Edmonton, AB, Canada
modayil@deepmind.com

Fatima Davelouis Gallardo
Department of Computing Science
University of Alberta / Amii
Edmonton, AB, Canada
daveloui@ualberta.ca

Michael Bowling
DeepMind &
Department of Computing Science
University of Alberta / Amii
Edmonton, AB, Canada
mbowling@ualberta.ca

Abstract

The performance of a reinforcement learning (RL) system depends on the computational architecture used to approximate a value function. We propose an online RL algorithm for adapting a value function's architecture and efficiently finding useful nonlinear features. The algorithm is evaluated in a spatial domain with high-dimensional, stochastic observations. Our method outperforms non-adaptive baseline architectures and approaches the performance of an architecture given side-channel information about observational structure. These results are a step towards scalable RL algorithms for more general problem settings, where observational structure is unavailable.

1 Introduction

Architectures for value function approximation typically impose sparse connections with prior knowledge of observational structure. When this structure is known, architectures such as convolutions, transformers, and graph neural networks can be inductively biased with fixed connections. However, there are times when observational structure will be unavailable or very difficult to encode as an architectural bias—for instance, relating sensors that are randomly dispersed in space. Yet in these situations it is still desirable to approximate value functions with a sparsely-connected architecture for computational efficiency. We explore the open question of whether useful representations can be constructed when observational structure is unknown—particularly in the incremental, online setting without access to a replay buffer.

Prior work has viewed the structure of observations as a hidden aspect of the environment [1]. Fixed architectural topologies have previously been used to relate inputs; these methods examined small input spaces, a combinatorially-large space of graphs [6], or offline methods that learn to reduce connections of a dense architecture [2]. Early work treated observational structure as the learning

*This extended abstract builds on the following article: "J. Martin, J. Modayil. Adapting the Function Approximation Architecture in Online Reinforcement Learning. CoRR abs/2106.09776, 2021"

†Equal contribution. Correspondence to John D. Martin

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

target: first positing a family of smooth topologies then selecting one to minimize a reconstruction loss [4].

We propose an online algorithm that adapts connections of a neural network using information deriving strictly from the reinforcement learning (RL) agent's experience stream, using many parallel auxiliary predictions. The algorithm is validated in a synthetic domain with high-dimensional stochastic observations. Results show the algorithm can adapt an approximation architecture without incurring substantial performance loss, while also remaining computationally tractable. Code has been made publicly available at `https://github.com/jdmartin86/frogseye`. Our key contributions are as follows.

Online adaptive architecture with reduced inductive bias: Using many parallel auxiliary learning objectives, our architecture dynamically connects observations to a set of filter banks to form useful nonlinear features.

Useful neighborhoods: The proposed algorithm is shown to compute sparse neighborhoods that perform comparably well to neighborhoods formed from side-channel distance information, and it substantially outperforms static baseline architectures.

2 Problem Setting

We are interested in the continual setting where the learner needs to adapt to changes in the environment in real time. More specifically, this work considers the standard RL prediction setting [10]. The return at time $t \in \mathbb{N}$ is the discounted sum of future rewards, $G_t \equiv R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$. The *value function* gives the expected return from a state: $v(s) \equiv \mathbb{E}[G_t | S_t = s]$. Instead of experiencing states directly, the learner receives a stream of observation vectors $\mathbf{o}_t \in \mathbb{R}^d$ and rewards. The learner's only knowledge of the environment state S_t and dynamics comes from this single stream of experience. With no direct access to the environment state, the learner forms an approximate value function to estimate the expected return. In RL, temporal difference (TD) learning is a commonly used online approximation technique — at each time step, the agent sees a new observation and reward, which are used to update state value estimates, as follows:

$$\hat{v}(\mathbf{o}_t) \leftarrow \hat{v}(\mathbf{o}_t) + \alpha \big[r_{t+1} + \gamma \hat{v}(\mathbf{o}_{t+1}) - \hat{v}(\mathbf{o}_t) \big]$$

where α denotes the step-size. Under a linear approximation, the value function is defined as a function of a feature vector $\mathbf{x}_t \in \mathbb{R}^\ell$, where

$$\hat{v}(\mathbf{x}_t; \mathbf{w}_t) \equiv \mathbf{w}_t^\top \mathbf{x}_t, \quad \hat{v}(\mathbf{x}_t; \mathbf{w}_t) \approx v(S_t). \quad (1)$$

Algorithm 1 Online Value Estimation with Prediction Adapted Neighborhoods

Initialize: $\mathbf{w}$, $\mathbf{z}$, $\mathbf{A}$ (fixed), $\mathbf{a}$ (fixed), $\mathbf{M}^{1:m}$, $\bar{\mathbf{w}}^{1:m}$, $\bar{\mathbf{z}}^{1:m}$.
Receive observation $\mathbf{o}_1$ from the environment.
$\mathbf{x}_1 \leftarrow \text{ComputeFeatures}(\mathbf{o}_1, \mathbf{M}^{1:m}, \mathbf{A}, \mathbf{a})$
for $t = 1, 2, 3, \cdots$ **do**
  Receive $r_{t+1}, \mathbf{o}_{t+1}$ from the environment.
  $\bar{\mathbf{x}}_j \leftarrow \mathbf{o}_j$ **for** $j \in \{t, t+1\}$
  parallel for $i \in \{1, \cdots, m\}$
    # Update GVF weights with TD(λ).
    $\bar{r}^i_{t+1} \leftarrow \mathbf{o}_{t+1}[c(i)]$
    $\delta \leftarrow \bar{r}^i_{t+1} + \gamma \bar{\mathbf{w}}^{i\top} \bar{\mathbf{x}}_{t+1} - \bar{\mathbf{w}}^{i\top} \bar{\mathbf{x}}_t$
    $\bar{\mathbf{z}}^i \leftarrow \gamma\lambda \bar{\mathbf{z}}^i + \bar{\mathbf{x}}_t$
    $\bar{\mathbf{w}}^i \leftarrow \bar{\mathbf{w}}^i + \bar{\alpha} \delta \bar{\mathbf{z}}^i$
    # Construct top-k selection matrix.
    $\boldsymbol{\ell} \leftarrow \text{Top}(k, \bar{\mathbf{w}}^i)$
    parallel for $j, l \in [1, k] \times [1, d]$
      $\mathbf{M}^i_{j,l} \leftarrow \mathbf{1}\{l = \boldsymbol{\ell}_j\}$
    $\mathbf{y}^i \leftarrow \mathbf{f}(\mathbf{A}\mathbf{M}^i \mathbf{o} + \mathbf{a})$
  $\mathbf{x}_{t+1} \leftarrow \text{concatenate}(\mathbf{o}, \mathbf{y}^1, ..., \mathbf{y}^m)$
  # Update main prediction weights with TD(λ).
  $\delta \leftarrow r_{t+1} + \gamma \mathbf{w}^\top \mathbf{x}_{t+1} - \mathbf{w}^\top \mathbf{x}_t$
  $\mathbf{z} \leftarrow \gamma\lambda \mathbf{z} + \mathbf{x}_t$
  $\mathbf{w} \leftarrow \mathbf{w} + \alpha \delta \mathbf{z}$

In this work the learner incrementally updates its weights $\mathbf{w}_t$ online with a TD algorithm.

2.1 An approximation architecture for the online setting:

We consider an architecture that computes nonlinear features from a sparsely-connected neural network with one hidden layer of random weights. A *neighborhood* is the set of inputs connected to a given nonlinear feature in the network; here it contains a sparse subset of the input, similar to an image patch used with convolutional architectures for vision. Nonlinear features from the i-th neighborhood are computed as a composition of three functions, $\mathbf{y}^i_t \equiv \mathbf{f}(\mathbf{A}\mathbf{M}^i \mathbf{o}_t + \mathbf{a})$. First is a neighborhood selection matrix $\mathbf{M}^i \in \{0, 1\}^{k \times d}$, then a linear projection $\mathbf{A} \in \mathbb{R}^{n \times k}$ shared between all the neighborhoods, and lastly a nonlinearity $\mathbf{f} \colon \mathbb{R}^n \to \mathbb{R}^n$. The neighborhood selection matrix $\mathbf{M}^i$

is an orthogonal rank-k matrix with one-hot columns—used to mask out an ordered selection of k elements of the observation. The linear projection $\mathbf{A}$ can be thought of as a set of filters, and $\mathbf{a} \in \mathbb{R}^n$ is a bias. The function $\mathbf{f}$ applies a fixed nonlinearity $f : \mathbb{R} \to \mathbb{R}$ to each element of its n-dimensional input: $\mathbf{f}(\mathbf{z}) = (f(\mathbf{z}_1), ..., f(\mathbf{z}_n))$. The full feature vector, $\mathbf{x}_t$, contains nonlinear features from m neighborhoods, $\mathbf{M}^i\mathbf{o}_t$, and the current observation, $\mathbf{x}_t \equiv \text{concatenate}(\mathbf{o}_t, \mathbf{y}_t^1, \ldots, \mathbf{y}_t^m)$.

3 Prediction Adapted Neighborhoods

We propose using information that derives from auxiliary RL predictions, specified as general value functions (GVFs) [11]. The resulting collections of observation subsets are called *prediction adapted neighborhoods*. This idea stems from the insights of previous works [8], [5], [9].

A GVF is defined as the expected return of some auxiliary reward signal $\bar{R}_{t+1}^i$, known as a *cumulant*. Here auxiliary rewards are given by observation components, and their returns are predicted under the same discount and policy as the main value function (1): $\bar{G}_t^i \equiv \bar{R}_{t+1}^i + \gamma \bar{R}_{t+2}^i + \gamma^2 \bar{R}_{t+3}^i + \ldots$. A selector function $c(i)$ returns an index into the observation vector to determine the i-th cumulant $\bar{r}_{t+1}^i \equiv \mathbf{o}_{t+1}[c(i)]$, for $i = 1, \cdots, m$. In this work, GVFs are approximated by linear functions of the observation $\bar{\mathbf{x}}_t \equiv \mathbf{o}_t$, with weights $\bar{\mathbf{w}}^i$: $\bar{\mathbf{w}}^{i\top} \bar{\mathbf{x}}_t \approx \mathbb{E}[\bar{G}_t^i | S_t = s]$.

One of our key discoveries is that observations can be related when used as auxiliary rewards for linear GVFs. Recall a GVF in this work predicts the future discounted sum of an observation signal. Also recall that a GVF is approximated with a linear combination of all observation components. Those components that closely relate to an auxiliary reward tend to have high-magnitude prediction weights. Based on this principle, prediction adapted neighborhoods are formed with observations with high absolute GVF weights $\bar{\mathbf{w}}^i$.

Algorithm 1 outlines how to compute prediction adapted neighborhoods in the online prediction setting (lines 6–14), with TD(λ) and accumulating traces $\mathbf{z}$. The algorithm constructs each neighborhood with the k observations whose absolute GVF weights are largest (lines 12–14), and it encodes them with the selection matrices $\mathbf{M}^i$.

In contrast to prior work that regularizes the internal architecture weights $\mathbf{A}$ with auxiliary prediction losses [3], our method uses auxiliary GVFs to impose sparse connections with a predictive structure. This information derives entirely from the observation stream, with no a priori knowledge of the observational structure. Our proposed algorithm continually adapts the architecture's connections in response to patterns of the observation stream.

4 Empirical Results in the Frog's Eye Domain

Drawing inspiration from the arrangement of light receptors in a frog's eye, we introduce an environment for studying continual prediction in the absence of observational structure (Figure 1). In our environment, light receptors have a uniformly-irregular spatial distribution. A simulated frog needs to anticipate the arrival of an insect without any knowledge of how its light receptors relate to one another.

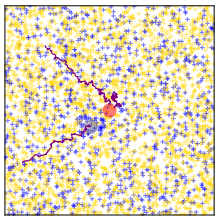

Figure 1: **The Frog's Eye domain:** An insect (gray circle) is detected by irregularly-distributed proximity receptors (blue: on, gold: off). Insect trajectories are shown in mauve. A reward of +1 is received upon entering the circular red region.

The full observation vector $\mathbf{o}_t \in \{0, 1\}^{4000}$ is given from a random ordering of 4000 proximity receptor outputs. These receptors are scattered randomly at the start of learning and then held fixed. Observations are corrupted with fifty-percent uniform binary noise. The reward is $+1$ whenever the insect enters a circular region at the center of the observable space, and it is zero otherwise. An insect entering the circular region will disappear and respawn at a random location. This process continues indefinitely.

Evaluation: Our experiments compare prediction error of different value function architectures while specifically controlling for the effects of neighborhood selection. All our baselines use TD(λ) for policy evaluation with state value functions. We compare empty neighborhoods with

$m = 0$ (denoted as Linear), sparse neighborhoods with k randomly-selected observation components (Random), prediction adapted neighborhoods from fixed randomly-selected cumulants (Adaptive), and sparse neighborhoods containing the k-nearest sensors to each cumulant using side-channel distance information (Distance). We also report the average and standard error confidence intervals from 30 trials, which were run to 5 million steps and observed to complete in under two hours on a V100 GPU. Results that used the ReLU nonlinearity are shown in Figure 2. Similar trends were also observed with the LTU and Majority filters. More experimental details including information about hyperparameter selection are provided in the full-length paper [7].

4.1 Prediction Adapted Neighborhoods are Useful

Our first experiment asks: *do prediction adapted neighborhoods provide measurable utility for approximating the main value function?* Figure 2 shows learning curves of prediction error. The Adaptive architecture—using prediction adapted neighborhoods—leads to little performance loss compared to the Distance architecture when evaluated across three types of activation functions. Recall the Distance architecture uses neighborhoods that contain the k-nearest sensors to each cumulant. The small performance gap between Adaptive and Distance suggests that prediction adapted neighborhoods are just as useful in this domain as neighborhoods biased with side-channel distance information.

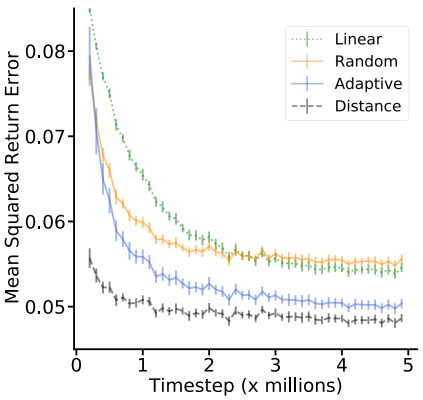

Figure 2: The Adaptive architecture (using prediction adapted neighborhoods) approaches the performance of Distance (biased with knowledge of observational structure). Adaptive also performs better than fixed architectures using random neighborhoods (Random) or none at all (Linear).

4.2 The Spatial Structure of Adapted Neighborhoods

A final inspection examined whether the prediction weights of a GVF contained any spatial structure. Figure 3 shows one set of auxiliary weights as learning progresses for a randomly-selected GVF. Clearly there are GVFs whose weights encode a local degree of spatial structure. Furthermore, this structure appears temporally stable over the extended regime of ten million time steps. These two points highlight that even without prior knowledge of the observation's spatial structure, auxiliary GVFs are able to relate observations in a similar way—ultimately one that is useful for the main prediction.

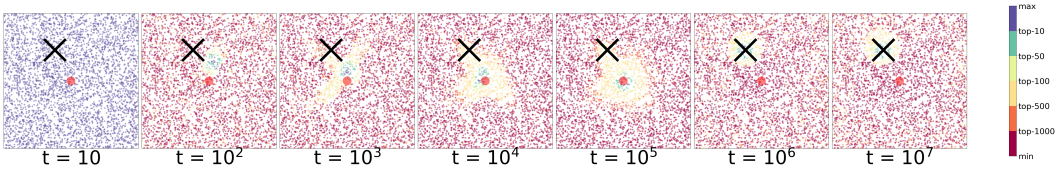

Figure 3: A spatial distribution of auxiliary weights is shown for a random GVF with a cumulant marked by an $\times$. The top ten neighborhood converges around the cumulant.

5 Conclusion

This paper addressed how an RL system could construct a value function architecture in the incremental online setting for prediction, and in the absence of observational structure. One of our key discoveries was that weights of auxiliary predictions could be used to relate observations and impose useful sparse connections in a random neural network approximating a value function. We believe this work could be useful for designing general RL systems that acquire knowledge from sensory inputs whose observational structure is unknown. Moreover, our work opens avenues that could allow designers to reduce the amount of architectural bias imposed. In future work, we would like to address some limitations of the current study; for instance, we would like to explore how to optimize the network's hidden weights online and in the RL control setting.

References

[1] R. Evans, J. Hernández-Orallo, J. Welbl, P. Kohli, and M. Sergot. Making sense of sensory input. *Artificial Intelligence*, 293:103438, 2021.

[2] T. Gale, E. Elsen, and S. Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

[3] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[4] N. Le Roux, Y. Bengio, P. Lamblin, M. Joliveau, and B. Kégl. Learning the 2-d topology of images. *Advances in Neural Information Processing Systems*, 20:841–848, 2007.

[5] M. L. Littman, R. S. Sutton, and S. P. Singh. Predictive representations of state. In *Advances in Neural Information Processing Systems 14*, pages 1555–1561, 2001.

[6] A. R. Mahmood and R. S. Sutton. Representation search through generate and test. In *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[7] J. D. Martin and J. Modayil. Adapting the function approximation architecture in online reinforcement learning. *arXiv preprint arXiv:2106.09776*, 2021.

[8] J. Modayil, A. White, and R. S. Sutton. Multi-timescale nexting in a reinforcement learning robot. *Adaptive Behavior*, 2014.

[9] D. Pierce and B. J. Kuipers. Map learning with uninterpreted sensors and effectors. *Artificial Intelligence*, 92(1-2):169–227, 1997.

[10] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[11] R. S. Sutton, J. Modayil, M. Delp, T. Degris, P. M. Pilarski, A. White, and D. Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 761–768, 2011.