WHAT SHOULD I KNOW? USING META-GRADIENT DESCENT FOR PREDICTIVE FEATURE DISCOVERY IN A SINGLE STREAM OF EXPERIENCE

Alexandra Kearney, Anna Koop
Department of Computing Science
University of Alberta
Edmonton, AB, Canada
{kearney,akoop}@ualberta.ca

Johannes Günther
Alberta Machine Intelligence Institute &
Department of Computing Science
University of Alberta
Edmonton, AB, Canada
gunther@ualberta.ca

Patrick M. Pilarski
Alberta Machine Intelligence Institute &
Department of Medicine &
Department of Computing Science
University of Alberta
Edmonton, AB, Canada
pilarski@ualberta.ca

ABSTRACT

In computational reinforcement learning, a growing body of work seeks to construct an agent's perception of the world through predictions of future sensations; predictions about environment observations are used as additional input features to enable better goal-directed decision-making. An open challenge in this line of work is determining from the infinitely many predictions that the agent could possibly make which predictions might best support decision-making. This challenge is especially apparent in continual learning problems where a single stream of experience is available to a singular agent. As a primary contribution, we introduce a meta-gradient descent process by which an agent learns 1) what predictions to make, 2) the estimates for its chosen predictions, and 3) how to use those estimates to generate policies that maximize future reward—all during a single ongoing process of continual learning. In this manuscript we consider predictions expressed as General Value Functions: temporally extended estimates of the accumulation of a future signal. We demonstrate that through interaction with the environment an agent can independently select predictions that resolve partial-observability, resulting in performance similar to expertly specified GVFs. By learning, rather than manually specifying these predictions, we enable the agent to identify useful predictions in a self-supervised manner, taking a step towards truly autonomous systems.

1 MAKING SENSE OF THE WORLD THROUGH PREDICTIONS

It has long been suggested that predictions of future experience can provide useful and intuitive features to support decision-making—particularly in partially-observable or non-Markovian environments (Singh et al., 2003; Littman et al., 2001; Jaeger, 2000). It is certainly true for biological agents: humans and many animals build predictive sensorimotor models of their world. These predictions of experience form the basis for biological perception (Rao & Ballard, 1999; Wolpert et al., 1995; Gilbert, 2009). A principled and well understood way of making temporally extended predictions in computational reinforcement learning is by estimating many value functions. Value functions predict the long-term expected accumulation of a signal in a given state (Sutton, 1988), and can predict not only reward, but any signal available to an agent via its senses (White, 2015). Prior works have used these General Value Function (GVF) estimates as features to adapt the control interfaces of bionic limbs (Edwards et al., 2016), design reflexive control systems for robots (Modayil & Sutton, 2014) and living cats (Dalrymple et al., 2020), and to inform industrial welding systems of estimated weld quality (Günther et al., 2016). In this paper, we explore how an agent can independently choose GVFs in order to augment its observations in order to construct it's own *agent-state*: an approximation of state from the agent's subjective perspective.

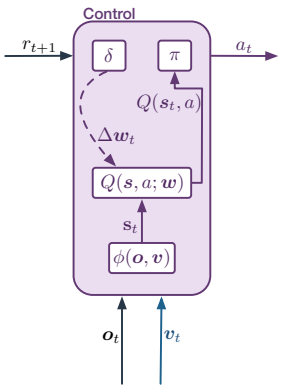
An open challenge when using GVF estimates as input features is determining what aspects of an agent's experience to predict. Of all the possible predictions an agent could make, which subset of GVFs are most useful to inform and support decision making? This choice is often made by the system designer (Modayil & Sutton, 2014; Dalrymple et al., 2020; Edwards et al., 2016; Günther et al., 2016). However, recent work has explored how an agent might independently specify its own GVFs. In Schlegel et al. (2018) an agent randomly selects the parameters that define what aspect of the environment is being estimated. After a period of learning, the agent replaces a subset of its GVFs based on their learning progress.

Determining which GVFs to replace is a core challenge for such *generate-and-test* approaches: A GVF may be accurate and have low prediction error, but just because a prediction is well estimated does not mean that it is useful as a predictive feature for control (Kearney et al., 0). Examining a learned prediction estimate without considering its use—as is done in generate-and-test for GVF specification—inherently limits the ability of an agent to choose *useful* predictive features.

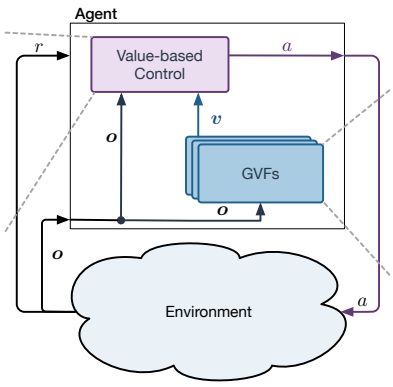
An alternative to random selection of GVFs is to parameterise the specification of a GVF and perform meta-gradient descent. By taking the gradient of a control learner's error with respect to a GVF's meta-parameters, what each prediction is about can be incrementally updated based on feedback from the control learner. Although not used for learning predictive inputs, recent work has shown early success in using meta-gradient descent as a means of learning meta-parameters that specify GVFs (Veeriah et al., 2019) for use as auxiliary tasks (Jaderberg et al., 2017). When used as auxiliary tasks, GVF estimates themselves are not directly used in decision-making, but rather as regularisers for the control agent's artificial neural network. In this auxiliary task setting, the parameters that determine what is being predicted and the parameters that are used to select actions are explicitly kept and learned independent of one another. We propose meta update where the core RL update of a control learner directly influences *what* an agent is predicting.

2 Meta-Learning General Value Functions

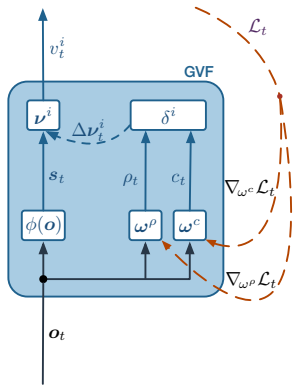
In this manuscript, we integrate the discovery and use of GVFs for reinforcement learning control problems. We present a fully self-supervised approach, using meta-gradient descent to autonomously discover GVFs that are useful as predictive features for control. We do so by parameterising the functions that determine what aspect of the environment a GVF prediction is about, and constructing a loss that shapes the predictions based on the control agent's learning process. The resulting learning process can be successfully implemented incrementally and online. By this process, agents are able to independently specify GVFs to be used directly as features by a control learner in order to solve two partially-observable problems. In doing so, we are providing a new solution to a long standing problem in using GVFs as predictive input features.



(a) The control agent updates its action-value function $Q(\boldsymbol{s}, a; \boldsymbol{w})$ according to the TD-error δ and chooses actions according to policy π based on value estimates..



(b) The typical agent-environment diagram of Reinforcement Learning, where the control agent learns values as a function of both observations and GVF outputs.



(c) A GVF outputs predictions according to its current weights $V(\boldsymbol{s}; \boldsymbol{\nu})$, while its updates are defined by the cumulant c, discount γ, and policy-correction ρ.

Figure 1

Our meta-learning process (Figure 1b) operates on an agent structured in three parts : 1) a value-based control unit that learns weights $\boldsymbol{w}$ for an action-value function (Figure 1a); 2) a collection of GVFs that each learn weights $\boldsymbol{\nu}$ to output prediction vector $\boldsymbol{v}$ (Figure 1c); and 3) a set of meta-weights $\boldsymbol{\omega}$ that parameterize each GVF's learning rule (the right half of Figure 1c).

The control unit is a typical Q-learning agent, although it learns a value function over the *agent-state* $\boldsymbol{s}$, which is constructed from the observations $\boldsymbol{o}$ and a vector of GVF predictions $\boldsymbol{v}$, rather than observation alone:

$$\boldsymbol{s}_t = \phi(\boldsymbol{o}_t, \boldsymbol{v}_t). \tag{1}$$

The action-value function $Q(\boldsymbol{s}_t, a; \boldsymbol{w}_t)$ may be learned by any relevant RL algorithm. We define the loss function $\mathcal{L}_t = \delta_t^2$, where:

$$\delta_t = r_{t+1} + \gamma \max_a Q(\boldsymbol{s}_{t+1}, a; \boldsymbol{w}_t) - Q(\boldsymbol{s}_t, a_t; \boldsymbol{w}_t). \tag{2}$$

GVFs may use any reinforcement learning method for learning, but their structure is defined by three functions of the current state[1]: the cumulant or target c, the discount or termination signal γ, and the policy π (see chapter 4 in White (2015)). The cumulant function $z(\boldsymbol{s}_t)$ determines the current target c_t: in classic RL, the cumulant simply singles out the current reward r_t. The discount factor γ determines how far into the future the signal-of-interest should be attended to. While in simple RL approaches it is generally a fixed value, in GVFs it can be any function of the current state $\gamma_t = g(\boldsymbol{s}_t)$. The policy allows each GVF to condition its prediction on specific behaviours, and is used to compute the importance-sampling correction against the behaviour policy μ: $\rho_t = \frac{\pi(\boldsymbol{s}_t, a_t)}{\mu(\boldsymbol{s}_t, a_t)}$. The output of a GVF is determined not only by the current weights $\boldsymbol{\nu}$ but also the functions that define its update procedure. We will use $\boldsymbol{\omega}$ to refer to the collective parameters that define the GVF structure, disambiguating with superscripts when necessary:

$$v_t^i = V(\boldsymbol{s}; \boldsymbol{\nu}^i, \boldsymbol{\omega}^i). \tag{3}$$

During execution of our meta-learning process, each time-step contains an action and learning phase. First, the agent receives an observation from the environment $\boldsymbol{o}_t$, which is used to compute the GVF state

$$\boldsymbol{s}_t^\nu = \phi^\nu(\boldsymbol{o}_t). \tag{4}$$

For each GVF i, the prediction value v_t^i is calculated as a function of the GVF state $\boldsymbol{s}_t^\nu$ and prediction weights $\boldsymbol{\nu}^i$:

$$v_t^i = V(\boldsymbol{s}_t^\nu; \boldsymbol{\nu}_t^i). \tag{5}$$

The vector of GVF predictions, along with the current observation, is transformed into the agent-state using a fixed function (see Equation 1 where ϕ may include eg: state aggregation, tile coding, artificial neural net). The policy unit π uses $Q(\boldsymbol{s}_t, a; \boldsymbol{w}_t)$ to determine the next action[2]. Once the action is executed and $(\boldsymbol{o}_{t+1}, r_{t+1})$ received, the learning phase begins.

The key to our meta-learning method is that the Q-learning error, as noted earlier (see Equation 2 and illustrated with the read lines in Figure 1c), is a function of not only the value function weights $\boldsymbol{w}$, but also the agent-state vector $\boldsymbol{s}$. As the agent-state is constructed from the GVF predictions, which in turn are are adjusted according to the meta-weights $\boldsymbol{\omega}$, we can use the control agent's loss function $\mathcal{L}_t = \delta_t^2$ to update the meta-weights. For the ith GVF, meta-weights $j \in \{c, \rho\}$ are adjusted:

$$\Delta\boldsymbol{\omega}_t^{i,j} = \alpha^{i,j} \nabla_{\omega^{i,j}} \mathcal{L}_t \tag{6}$$

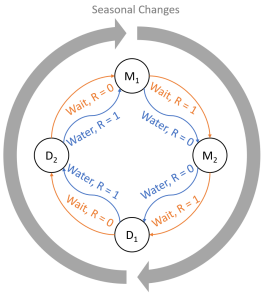
Once the meta-weights have been updated, each GVF computes the current ρ_t, c_t, and γ_t, updates its predictions weights $\boldsymbol{\nu}$. Finally, updated agent-state $\boldsymbol{v}_{t+1}$ is computed and used by the control agent to update its Q-learning weights $\boldsymbol{w}_{t+1}$. Pseudocode is provided in Appendix C.

3 Can An Agent Learn What To Predict?

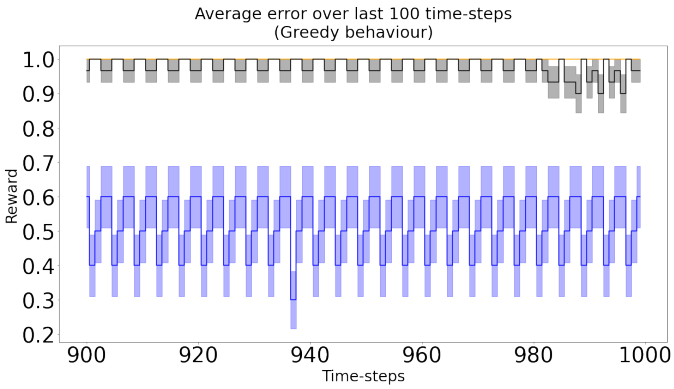
Using the meta process we introduced, can an agent find useful GVFs for use as predictive input features? We first evaluate meta specification of GVFs on a partially observable control problem, Monsoon World (Figure 2a). We choose to introduce this problem as it is a minimal, clear example of a situation where temporal abstraction is neccessary to solve the problem. This enables us to assess in a clear way whether meta-gradient descent (MGD) is capable of specifying useful predictions, and precisely examine what predictions the agent chooses to learn.

[1] note that this is the GVF-specific state, independent of the *agent-state* defined above

[2] in the following results we use ϵ-greedy action selection



(a) The monsoon problem. On each time-step the agent observes a binary value that determines whether the crops have grown. Growth of crops is determined by both the action an agent takes (to water crops or not), and the unobserved underlying season. There are four phases of the season that an agent can exist in: two monsoon and two drought (inner circles). The outer arrows indicate how the seasons change as the agent transitions through the cycle.



(b) Three different learners that use 1) the environmental observations as inputs (blue), 2) two additional inputs which express the seasons (in orange), 3) two additional predictions that are updated using meta-gradient descent (in black). Each independent agent's mean reward is averaged over 30 independent trials. Error bars are standard error.

Figure 2

In Monsoon World, there are two seasons: monsoon and drought. The underlying season determines whether the agent receives reward for its chosen action, however the underlying season is not directly observable. Although the agent cannot directly observe seasons, it can observe the result of a given action: something impacted by the seasons. The agent tends to a field by choosing to either water, or not water their farm. Watering the field during a drought will result in a reward of 1; watering the field during monsoon season does not produce growth and results in a reward of 0, and vice versa during a monsoon. If the agent chooses the right action corresponding to the underlying season, a reward of 1 can be obtained on each time-step. Regardless of the action chosen by the agent, time progresses.

This monsoon world problem can be solved, if the agent reliably estimates how long until watering produces a particular result. This can be done by learning *echo GVFs* (c.f. (Schlegel et al., 2021)). Echo GVFs estimate the time to an event using a state-conditioned discount and cumulant. In plain terms, the two GVF's that when learned provide estimates that solve the problem can be described as "How long until watering produces growth"? or "How long until not watering produces growth"? Indirectly, these capture the time until either the monsoon or drought. These two predictions can be described as *off-policy* estimates: predictions that are conditioned on a particular behaviour. Given the agent has two actions where a_0 is not watering and a_1 is watering, we can describe the policy "if the agent waters" as a deterministic policy $\pi = [0, 1]$. The signal of interest is, $c_t = 1$ if $r_t = 1$ & 0 otherwise. Similarly, a state-dependent discounting function terminates the accumulation, $\gamma_t = 0$ if $c_t = 1$ & 0.9 otherwise. Off-policy GVFs can be estimated online, incrementally, while the agent is engaging in behaviours that do not strictly match the target policies of the prediction (Maei, 2011). Having constructed the aforementioned GVFs, an agent can express what is hidden from its observation stream: how long until the next season. While no information was given about the seasons, the agent is able to learn about the season by relating what is sensed by the agent with the actions that were taken.

3.1 Learning to Specify GVFs in Monsoon World

GVF estimates can resolve the partial observability of monsoon world. Through MGD, can an agent find similar predictions? We compare three different agent configurations (Figure 2b): 1) a baseline agent that only receives environmental observations as inputs (in blue), 2) an agent that in addition to the environmental observations, two inputs that capture underlying seasons ('oracle', in orange), and 3) an agent that has two GVFs whose cumulants and policies are learned through meta-gradient descent (in black).

Learning by MGD to specify GVFs introduces two additional sets of meta-weights to initialise: weights $\boldsymbol{\omega}^\pi$ that specify the policy a prediction is condition on, and weights $\boldsymbol{\omega}^c$ that determine the signal of interest from the environment that is being learnt about. Target policies are initialised to an equiprobable weighting of actions, and are passed through a Softmax activation function $v_t^i = \texttt{softmax}(\boldsymbol{o}_t^\top \boldsymbol{w}_t^\pi)$ so that their sum is between [1, 0]. The cumulant meta-weights $\boldsymbol{\omega}^c$ are initialised to -5, and a sigmoid activation is applied such that $v^c = \texttt{sigmoid}(\boldsymbol{o}_t^\top \boldsymbol{\omega}^c)$. We pass the cumulant meta-weights through a sigmoid to bound the cumulant between [0,1]. The meta-weights are updated each time-step

incrementally. We apply an L2 regulariser to the meta-loss with $\lambda = 0.001$. The control learner is a linear Q-learner. Additional experiment details are in Appendix A.

In Figure 2b, we plot the average reward per time-step during the final 100 time-steps during which we evaluate agent performance given greedy behaviour. While the agents deterministically follow their policy during the evaluation phase, learning still occurs during the evaluation phase: updates are made to the GVFs, which affect the input observations to the control agent (in the case of the MGD agent), and the control learner continues to update its action-value function. This continued learning accounts for irregularity in the oscillations.

The policy learnt using only environment observations is roughly equivalent to equiprobably choosing an action: the learned policy is no better than a coin-toss (Figure 2b, depicted in blue). This is as expected, given observations alone are insufficient to determine the optimal action on a given time-step. When the underlying season is provided as input (depicted in orange), the learned policy is approximately optimal. By augmenting environmental observations with predictive features that estimate the time to each season, an agent is able to solve the problem. Using meta-gradient descent, the agent was able to select its own predictive features without any prior knowledge of the domain. *Using meta-gradient descent, the agent is able to solve the task with performance on-par with the hand-crafted solution without being given what to predict.*

If an agent can find GVFs to solve monsoon world, what are the useful predictions the agent found? Do they relate to our expert's GVFs? In Figure 3a, the value-estimates on the final 10 time-steps of each run are presented, as well as the meta-weights for the cumulants (Figure 3b) and policies (Figure 3c). We found that two distinct types of policy and cumulant were learned for each of the two GVFs. In one of the 30 independent trials, the agent failed to solve the problem, and we depict the learned meta-weights of this failed trial independently. The third column illustrates how different the relationship of the two value estimates are in this failure case (from both successful and expert-specified GVFs).

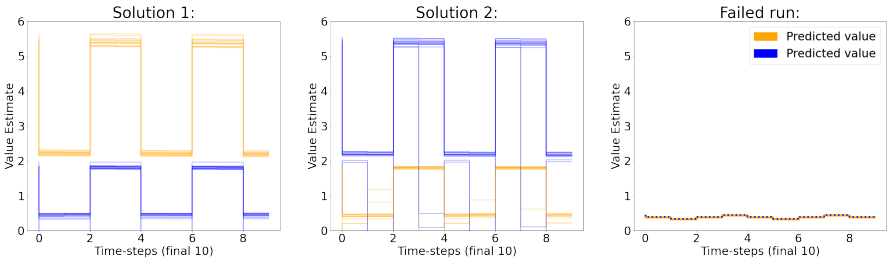
Over the course of successful runs, GVFs found by MGD do not look exactly like the echo GVFs introduced in Section 3. Some characteristics are similar: i.e., one of the policies approaches $\pi \approx [1, 0]$; however, the other policy $\pi \approx [0.8, 0.2]$ looks quite different from the deterministic policies. Even when the *value estimates* output by the self-supervised GVFs are similar to those from expert-specified GVFs, the cumulant and policies can be quite different.

For the best parameter settings in our sweep, one of 30 independent trials failed, achieving an average reward per-step of 0.5 (note this performance is similar to that of the agent with only environmental observations). For this failed run, the learned policy and cumulant do not fit the categorisations of cumulants or policies learned in successful trials. Importantly, the learned value estimates presented to the control agent as features do not share the same cyclic values that capture the underlying seasons of the environment. From this failed run, we can see that simply adding *any* prediction does not enable the agent to solve the problem: in successful trials, the policies and cumulants learned by MGD are meaningful and specific to the environment. These specific cumulants and policies are what enable the agents to solve the problem.

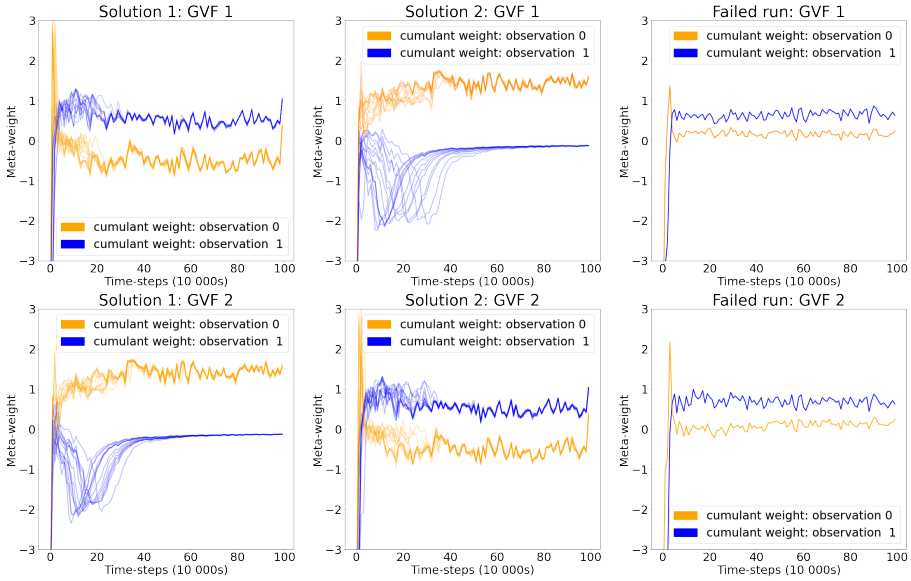
3.2 Learning to Specify GVFs in Frost Hollow

The previous example explored whether using MGD an agent could learn to specify predictions in order to resolve the partial-observability of its environment. In this section, we add two additional complications: 1) instead of a linear control agent, we use a more complex function approximator; and 2) we use a domain where the reward is sparse, thus complicating the GVF specification process. Frost Hollow (depicted in Figure 4) (Butcher et al., 2022; Brenneis et al., 2021) was first proposed as a joint-action problem where two agents attempt to cooperatively solve a control problem: a prediction agent passes a cue to a control agent based on a learned prediction; using this cue as an additional input, the control agent takes an action. Frost Hollow was designed as an environment where predictive inputs are used to augment a control-agent's inputs, making it well suited for assessing whether via MGD an agent can independently choose what GVFs to learn in order to improve performance on a control task.

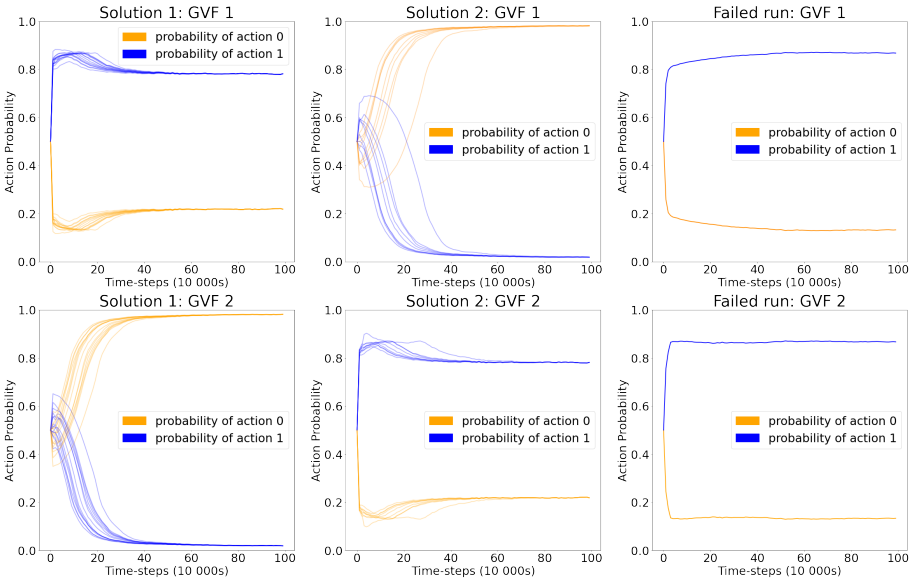
While simple, Frost Hollow poses a difficult learning problem. The reward is sparse: an agent can only observe a reward after successfully accumulating heat and dodging regular hazards. It takes at a minimum 50 time-steps, or 5 successive cycles of dodging the hazard successfully, before the agent can acquire a single reward. While the hazard itself is observable to the agent when active, the agent must pre-emptively take shelter before the hazard's onset in order to avoid losing its accumulated heat. All of this must be learnt from a sparse reward signal. Learning an additional GVF and using its estimate as an additional predictive feature can enable both humans, and value-based agents to successfully gain reward (Butcher et al., 2022; Brenneis et al., 2021).



(a) Value estimates from each GVF during all independent trials.



(b) The cumulant meta-weights $\boldsymbol{\omega}^c$ learned through MGD.



(c) The policy meta-weights $\boldsymbol{\omega}^\rho$ learned through MGD.

Figure 3

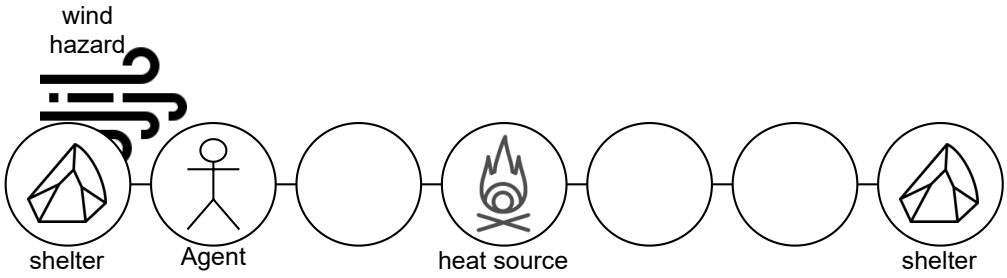
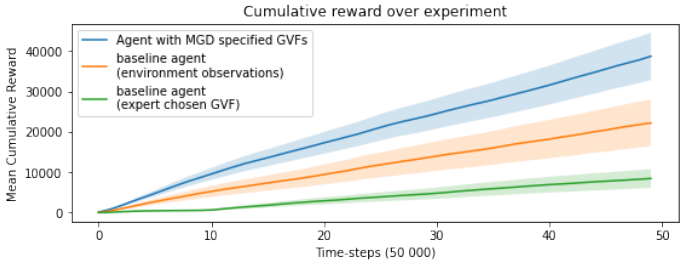


Figure 4: A depiction of the frost hollow problem. Frost hollow is a linear walk where an agent collects a unit of heat by standing next to a fire in the center state. Once the agent accumulates 12 units of heat, it receives a reward of 1. Every 8 time-steps, a wind hazard gusts for two consecutive time-steps, removing all of the agent's accumulated heat if it is exposed to the hazard. To avoid losing its heat, the agent can take shelter in either of the end states. On each time-step, the agent observes its own location, the amount of heat it has accumulated, and whether the wind hazard is present.

In the frost-hollow setting, the control agent receives a single *on-policy* prediction as an additional input feature: the GVF is conditioned on the agent's behaviour rather than a policy specified by MGD. The discount γ is a constant valued 0.9, following Butcher et al. (2022). We compare the meta-gradient learning process to two baselines: 1) where the control agent receives a expert defined GVF, as specified in Butcher et al. (2022); and where an agent that receives only the environment observations—no additional predictive features or information. In this setting we use a DQN agent for the control learner (Mnih et al., 2015) adapted from Dopamine (Castro et al., 2018). We train all agents for 249 000 time-steps, and evaluate their performance on the final 1000 time-steps. During the evaluation phase the ϵ is set to 0.01, limiting non-greedy actions. All reported results are averaged over 30 independent trials. See Appendix B for additional experiment details.

In Figure 5b, we report the average cumulative reward during the final evaluation steps. If an agent is deterministically following the optimal policy during the evaluation phase, the maximum possible cumulative reward is 50. In Butcher et al. (2022), the best performing agent with a highly specialised representation was able to achieve a cumulative reward of around 40; however, many of the agents with hand-selected predictions introduced in Butcher et al. (2022) received a cumulative reward of approximately 30.

In our experiments, the baseline agent without any additional predictions achieves an average cumulative reward of 7. By adding an additional predictive input feature that is specified by MGD, we are able to achieve an average cumulative reward of 18.7. Interestingly, the MGD agent learned to specify a cumulant that is different from those chosen in Butcher et al. (2022). Successful runs learn a cumulant that predominantly weights the accumulated heat, input 8 (as depicted in Figure 6). In Figure 6 we report the average meta-weights specifying the cumulant over the course of the entire experiment. In Butcher et al. (2022), the expert chosen prediction is of the oncoming hazard: input 7. There is a logic to the prediction selected by MGD: the heat an agent accumulates is directly related to



(a) Mean cumulative reward over the entire duration of the trials in Frost Hollow. Assuming an agent follows the optimal policy deterministically for the total 2.5 milllion time-steps the maximum possible reward is 125000 Error bars are the standard error.

Agent Type	**Cumulative Reward: Evaluation Phase**
baseline (environment obs)	7 ± 2.9
baseline (expert chosen GVF)	3.3 ± 1.6
Agent with MGD specified GVF	18.7 ± 4.2

(b) Cumulative reward and standard error of the mean during final 1000 evaluation steps for best configuration of each agent. Maximum possible cumulative reward is 50.

Figure 5: Mean cumulative reward for each agent in the Frost Hollow, averaged over 30 trials.

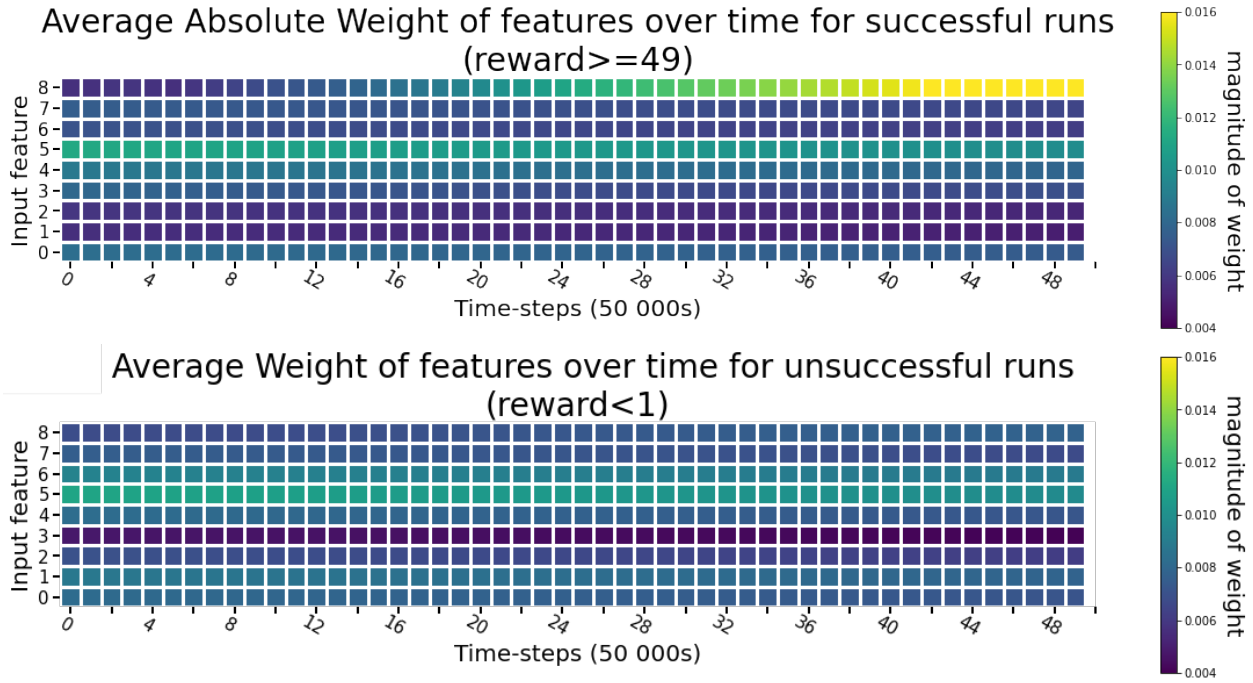


Figure 6: A depiction of the weights of the meta-weights ω_c for cumulants learned. Inputs $0-6$ are a binary encoding of the agent's location in the linear walk. Input 7 is a binary feature that encodes whether the wind hazard is present. Input 8 is the accumulated heat.

reward—reward is recieved after an agent acquires 12 heat. Moreover, the heat accumulated is indirectly related to the hazard: if the agent is unprotected before the hazard, it should anticipate a drop in it's accumulated heat. By predicting accumulated heat, the agent is dually capturing information about both the sparse reward signal and the original aspect of the environment that the expertly defined prediction sought to predict.

Interestingly, the baseline agent which used an expert-specified prediction performed less well than the MGD agent, receiving an average cumulative reward of 3.3. This is a revealing example: while the expert-specified GVF was well suited to the tabular setting explored in prior work (Butcher et al., 2022), its effectiveness as a predictive feature did not generalise to the function approximation setting. This highlights a challenge that we believe is present across domains and environmental settings. What predictive features might be useful to an agent is not only influenced by the environment, but also differences in state construction and the underlying learning method. Together, these factors all influence what GVFs may be useful for decision-making.

4 Limitations & Future Work

In this paper we demonstrate for the first time that end-to-end learning of GVFs used as predictive inputs is possible. Moreover, we demonstrate this within the setting that GVFs were originally proposed: continual life-long learning. This has been an open challenge in GVF research since their introduction 10 years ago. However, this paper is not the final word on MGD discovery of GVFs. Assessing how well the method presented scales in environments with non-stationarity, or greater observational complexity is important future work. In this paper, we decided to fix the discount γ—the time-horizon over which a prediction is estimated. Previous work has enabled GVFs to be learned over many timescales at once, enabling inference over arbitrary horizons (Sherstan et al., 2020). Future work could explore how Sherstan *et al.*'s γ-net formulation could improve the scalability and flexibilty of the meta-learning process we introduce. This paper makes progress in enabling agents to choose what to predict. How an agent may decide the number of predictions to learn remains an important open question. Similarly, how an agent might incrementally increase its capacity by adding new predictions during life-long learning remains to be explored. One possibility is to arrange predictions in multiple layers, similar to GVF networks: (Schlegel et al., 2021). Questions regarding GVF structure and scale are exciting open frontiers, and this manuscript provides a foundation that enables such questions to be asked in future work.

5 CONCLUSION

In this paper we introduced a process that enables agents to meta-learn the specifications of predictions in the form of GVFs. This process enabled agents to learn what aspects of the environment to predict while also learning the predictions themselves and learning how to use said predictions to inform action-selection. This meta-learning process was able to be implemented in an online, incremental fashion, making it possible for long-lived continual learning agents to self-supervise the specification and learning of their own GVFs. We found that an agent with no prior knowledge of the environment was able to select predictions that yielded performance equitable to, or better than agents using expertly chosen predictive features. Even in an environment with a sparse reward, an agent was still able to learn to specify useful predictions to use as additional features based on the control-learner's error. In sum total, this work provides an important step-change in the design of agents that use predictive knowledge.

It has long been suggested that predictions of future experience in the form of GVFs can provide useful features to support decision-making in computational reinforcement learning. Requiring system designers to re specify the GVFs for every permutation of an agent and its environment is a burden for applications of predictive features, and yet it is still the norm in both research and applied settings. Fundamentally, the requirement of designers to hand-specify GVFs prevents the development of fully independent and autonomous agents. Moreover, this has inhibited the application of GVFs, especially in long-lived continual learning domains that may exhibit non-stationarity. In this paper, we contributed a meta-gradient descent processes by which agents were able to find GVFs that improved decision-making relative to environment observations alone In doing so, we provide a research path for resolving an open challenge in the GVF literature that has existed for over a decade.

REFERENCES

Dylan J. A. Brenneis, Adam S. Parker, Michael Bradley Johanson, Andrew Butcher, Elnaz Davoodi, Leslie Acker, Matthew M. Botvinick, Joseph Modayil, Adam White, and Patrick M. Pilarski. Assessing human interaction in virtual reality with continually learning prediction agents based on reinforcement learning algorithms: A pilot study. *arXiv:2112.07774 [cs.AI]*, 2021.

Andrew Butcher, Michael Bradley Johanson, Elnaz Davoodi, Dylan J. A. Brenneis, Leslie Acker, Adam S. R. Parker, Adam White, Joseph Modayil, and Patrick M. Pilarski. Pavlovian signalling with general value functions in agent-agent temporal decision making. *arXiv:2201.03709 [cs.AI]*, 2022.

Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G. Bellemare. Dopamine: A Research Framework for Deep Reinforcement Learning. 2018. URL http://arxiv.org/abs/1812.06110.

Ashley N Dalrymple, David A Roszko, Richard S Sutton, and Vivian K Mushahwar. Pavlovian control of intraspinal microstimulation to produce over-ground walking. *Journal of neural engineering*, 17(3):036002, 2020.

Ann L Edwards, Michael R Dawson, Jacqueline S Hebert, Craig Sherstan, Richard S Sutton, K Ming Chan, and Patrick M Pilarski. Application of real-time machine learning to myoelectric prosthesis control: A case series in adaptive switching. *Prosthetics and orthotics international*, 40(5):573–581, 2016.

Daniel Gilbert. *Stumbling on happiness*. Vintage Canada, 2009.

Johannes Günther, Patrick M. Pilarski, Gerhard Helfrich, Hao Shen, and Klaus Diepold. Intelligent laser welding through representation, prediction, and control learning: An architecture with deep neural networks and reinforcement learning. *Mechatronics*, 34:1–11, 2016. ISSN 0957-4158. doi: https://doi.org/10.1016/j.mechatronics.2015.09.004. URL https://www.sciencedirect.com/science/article/pii/S0957415815001555.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *proceedings of the 5th International Conference on Learning Representations, ICLR 2017*, 2017.

Herbert Jaeger. Observable operator models for discrete stochastic time series. *Neural computation*, 12(6):1371–1398, 2000.

Alex Kearney, Anna J Koop, and Patrick M Pilarski. What's a good prediction? challenges in evaluating an agent's knowledge. *Adaptive Behavior*, 0(0):10597123221095880, 0. doi: 10.1177/10597123221095880. URL https://doi.org/10.1177/10597123221095880.

Michael L Littman, Richard S Sutton, and Satinder P Singh. Predictive representations of state. In *Advances in neural information processing systems 14: Proceedings of the 2001 Conference*, volume 14, pp. 30, 2001.

Hamid Reza Maei. *Gradient Temporal-difference Learning Algorithms*. PhD thesis, University of Alberta, 2011.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Joseph Modayil and Richard S Sutton. Prediction driven behavior: Learning predictions that drive fixed responses. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87, 1999.

Matthew Schlegel, Adam White, and Martha White. A baseline of discovery for general value function networks under partial observability. In *NeurIPS Workshop on Reinforcement Learning under Partial Observability: Montreal, Canada*, 2018.

Matthew Schlegel, Andrew Jacobsen, Zaheer Abbas, Andrew Patterson, Adam White, and Martha White. General value function networks. *Journal of Artificial Intelligence Research*, 70:497–543, 2021.

Craig Sherstan, Shibhansh Dohare, James MacGlashan, Johannes Günther, and Patrick M Pilarski. Gamma-nets: Generalizing value estimation over timescale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5717–5725, 2020.

Satinder P Singh, Michael L Littman, Nicholas K Jong, David Pardoe, and Peter Stone. Learning predictive state representations. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 712–719, 2003.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

Vivek Veeriah, Matteo Hessel, Zhongwen Xu, Janarthanan Rajendran, Richard L Lewis, Junhyuk Oh, Hado P van Hasselt, David Silver, and Satinder Singh. Discovery of useful questions as auxiliary tasks. *Advances in Neural Information Processing Systems*, 32, 2019.

Adam White. *Developing a predictive approach to knowledge*. PhD thesis, University of Alberta, 2015.

Daniel M Wolpert, Zoubin Ghahramani, and Michael I Jordan. An internal model for sensorimotor integration. *Science*, 269(5232):1880–1882, 1995.

A EXPERIMENT DETAILS: MONSOON WORLD

Monsoon world experiments ran for a total of one million time-steps. Each agent had a training phase of 990,000 time-steps. The agent's performance is evaluated during the final 1000 time-steps where ϵ is set to 0 and actions are chosen greedily so that we can compare average reward achieved from following the learned policies.

Monsoon world experiments are not run-time sensitive.

A.1 FUNCTION APPROXIMATORS AND STATE AGGREGATION

We use different function approximators to transform the given inputs to an *agent-state* $\boldsymbol{s}_t = \phi(\boldsymbol{o}_t, \boldsymbol{v}_t)$. Echo GVF estimates are in log-space (c.f. Schlegel et al. (2021) for more information on echo GVFs). Before using an echo GVF's estimate as inputs, we apply a log transformation to them (see Algorithm 7a).

A simple state-aggregation algorithm, `agg`is given in Algorithm 7b). We will abbreviate linear function approximation as `lin`.

a Log transform of prediction estimates

```
# Where v holds the value estimate from n GVFs.
# Where max γ_t is determined by the g function.

procedure TRANSFORM(v)
    v ← clip(log(v)/ log(max γ_t), 0, 1)
    return v
end procedure
```

b State aggregation of predictions, `agg`

```
# Where v are the value estimates from n GVFs.
# Where v^i < 10
# Where memsize is the allocated length for the binary feature vector.

procedure STATE(v)
    s = zeros(memsize)
    i ← ⌊v_0 + v_1 * 10⌋
    s^i = 1
    return s
end procedure
```

Figure 7: Standard function approximation steps for Echo GVFs

We compared three agents in the Monsoon environment: the agent that learns a value solely over the observations, **obs**; the agent that learns over both observations and expert-specified GVFs, **exp**; and the agent that learns over both observations and GVFs discovered through meta-gradient descent, **MGD**.

Each agent has a specific function-approximator for each of its component. All RL agents have a function approximator used by the control unit, ϕ. A GVF-based agent also has a choice of function approximator for the GVF predictions, ϕ^ν. Finally, the meta-gradient agents may also have a function approximator used for the meta-parameter update.

Parameters were chosen by performing a sweep across different values, choosing the best performing combination for each agent during the final 1000 evaluation steps in the experiment.

A.2 META-PARAMETER SPECIFICATION

The GVF's policy π is a fixed policy: the meta-weights determine the policy a GVF is conditioned on, but they are not a function of the observations: $\pi \leftarrow \text{softmax}(\boldsymbol{\omega}^\pi)$

Agent	ϕ	ϕ^ν	ϕ^ω
obs	`agg`	`agg`	-
exp	`agg`	`agg`	-
MGD	`lin`	`agg`	`lin`

(a) Function approximators used for each agent component.

Agent	ϵ	α^Q	α^V	α^ρ	α^c
obs	0.1	0.01	0.1	-	-
exp	0.1	0.01	0.1	n/a	n/a
MGD	0.5	0.0001	0.1	0.001	0.1

(b) Hyperparameters for each agent.

Figure 8: Parameter settings for different agent configurations

The cumulant c is a function of the observations, $c_t = \sigma(\boldsymbol{o}_t^\top \boldsymbol{\omega}^c)$, where $\boldsymbol{\omega}^c$ are the meta-weights for the cumulant, and $\boldsymbol{o}_t$ is the present environment observation.

B Experiment Details: Frost Hollow

All independent runs lasted 2,500,000 time-steps. Each agent had a training phase of 2,499,000 time-steps, and an evaluation phase in the final 1000 time-steps where $\epsilon = 0.001$ for ϵ-greedy action selection.

B.1 Function Approximators

As in the Monsoon World experiments, the meta-weights that specify the cumulants are a linear combination of features and weights with no activation function. For a single GVF the cumulant at time t is: $c_t = \boldsymbol{\omega}_t^\top \boldsymbol{o}_t$.

The GVF in the frost-hollow experiment uses a *bit-cascade* representation, as introduced in (Butcher et al., 2022).

The control agent uses an artificial neural network as its function approximator. In particular, `JaxDQNAgent` from Castro et al. (2018) and use the default parameter configurations for all control agents.

B.2 Parameter Settings

The DQN configuration uses the default parameter settings of Castro et al. (2018) for the `JaxDQNAgent`.

Additional reported parameters were chosen by performing a sweep across different values, choosing the best performing combination for each agent during the final 1000 evaluation steps in the experiment.

Agent	α^V	α^Q
obs	n/a	n/a
exp	0.001	n/a
MGD	0.001	0.0001

Table 1: Best parameter settings for different agent configurations

C Detailed pseudo-code for Meta-gradient Reinforcement Learning

Algorithm 1 A meta-gradient approach to self-supervised predictive Reinforcement Learning.

Choose hyperparameters:
Choose control agent's state function approximator ϕ, learning rate α, L2 regularizer λ and exploration rate ϵ.
Choose GVF state function approximator ϕ^{ν}, eligibility trace factor λ^{ν} and learning rate α^{ν} for GVF updates.
Choose α^{c}, α^{ρ} for meta-learning updates.

Initialise:
Initialise Q-learning weights $\boldsymbol{w}_0$
for all $i \in$ GVFs **do**
    Initialise GVF prediction v_0^i
    Initialise weights $\boldsymbol{\nu}_0^i$ and $\boldsymbol{\omega}_0^i$
end for

BEGIN
Receive initial observation $\boldsymbol{o}_0$
Calculate initial agent-state vector $\boldsymbol{v}_0$

for $t = 1 \ldots$ **do**
    # Control Agent
    Compute agent-state $\boldsymbol{s}_{t-1} \leftarrow \phi(\boldsymbol{o}_{t-1}, \boldsymbol{v}_{t-1})$
    Use action-values $Q(\boldsymbol{s}_{t-1}, a; \boldsymbol{w}_{t-1})$ and control policy π to choose action a_{t-1}
    Take action a_{t-1}, observe r_t and $\boldsymbol{o}_t$

    for $i \in$ GVFs **do**
        # Meta-gradient Update
        for each parameterized GVF component $j \in \{c, \rho\}$ **do**
            Update meta-weights $\boldsymbol{\omega}_t^j \leftarrow \boldsymbol{\omega}_{t-1} - \alpha^j \nabla_{\boldsymbol{\omega}^j} \mathcal{L}_t$
        end for

        # GVF Value Update
        Calculate ρ_t^i, c_t^i, γ_t^i using $\boldsymbol{\omega}_t^i$
        Update GVF weights $\boldsymbol{\nu}_t^i$ according to GVF learning procedure.
    end for

    # Control Agent Update
    Compute current agent-state $\boldsymbol{s}_t \leftarrow \phi(\boldsymbol{o}_t, \boldsymbol{v}_t)$
    Update agent weights $\boldsymbol{w}_t$ according to control agent learning procedure.
end for