Finding Useful Predictions by Meta-gradient Descent to Improve Decision-making

Alex Kearney
Department of Computing Science
University of Alberta
Edmonton, AB, Canada
hi@alexkearney.com

Anna Koop
Department of Computing Science
University of Alberta
Edmonton, AB, Canada
akoop@ualberta.ca

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

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

Abstract

In computational reinforcement learning, a growing body of work seeks to express an agent's model of the world through predictions about future sensations. In this manuscript we focus on predictions expressed as General Value Functions: temporally extended estimates of the accumulation of a future signal. One challenge is determining from the infinitely many predictions that the agent could possibly make which might support decision-making. In this work, we contribute a meta-gradient descent method by which an agent can directly specify what predictions it learns, independent of designer instruction. To that end, we introduce a partially observable domain suited to this investigation. We then demonstrate that through interaction with the environment an agent can independently select predictions that resolve the partial-observability, resulting in performance similar to expertly chosen value functions. 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 is often useful to break a challenging problem into sub-problems: progress on sub-tasks can support an agent's progress on a greater task, e.g., learning the values of states in order to approximate the optimal policy, or learning models of the world to better plan. One way an agent can create sub-problems and a world model is by learning predictions of its world—biological agents do this by building predictive sensorimotor models of their world (Rao and Ballard, 1999; Wolpert et al., 1995; Gilbert, 2009). One principled and well understood way of making temporally extended predictions in reinforcement learning is by learning and maintaining 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 (Sutton et al., 2011). Prior works have used general value estimates as features to adapt the control interfaces of bionic limbs (Edwards et al., 2016), design reflexive control systems for robots (Modayil and Sutton, 2014) and living cats

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

(Dalrymple et al., 2020), and to inform industrial welding about the process quality (Günther et al., 2016).

An open challenge when using GVFs is determining what to predict. Of all the possible predictions to make, which subset is most useful to inform and support decision making? This choice is typically made by the human designer of the system. However, previous work has used generate and test to choose which predictions are maintained, and which should be replaced (Schlegel et al., 2018). One hindrance of this method is the generator used to pick new predictions: the agent must explore a space of infinite predictions if they are chosen randomly. Moreover, common evaluation methods are not always reliable and can have adverse impacts on performance (Kearney et al., 2021). Recent work has explored meta-gradient descent as a means of learning meta-parameters that specify the predictions (Veeriah et al., 2019); however, in this case the estimates were used as auxiliary tasks—the estimates themselves were not directly used in decision-making.

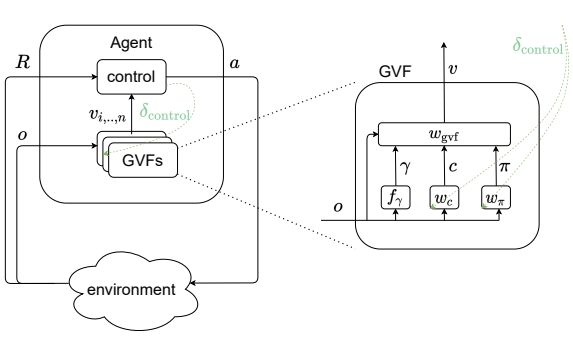
In this manuscript we propose a method of using meta-gradient descent to discover GVFs independent of human instruction and supervision. We do so by constructing a loss that shapes what the underlying predictions are about based on the control agent's learning process. All learning methods are updated incrementally and online. These value estimates can then be used directly as features by a control learner to solve a partially-observable problem.

2 Learning What to Predict: An Architectural Proposal

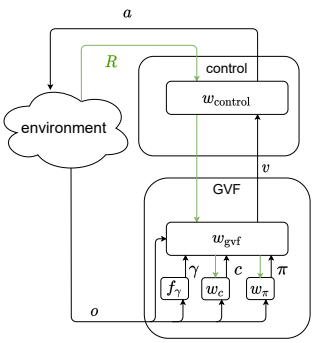
Our agent is structured in three parts (Figure 1): 1) a control learner that chooses each action; 2) a collection of GVFs that approximate value-functions specified by some policies $\pi_{1\cdots n}$, cumulants c, and discount functions $\gamma_{1\cdots n}$; and 3) meta-weights that determine the policy π (w_π) and cumulant c (w_c) each GVF is conditioned on. This architecture is shown in Figure 1; pseudocode describes the relationships between these components in Algorithm 1.

On each time-step, the agent observes the current state of the environment o_t. A collection of n GVFs perform a temporal difference update based on this observation, and prediction estimates are produced $v_{1\cdots n}$. These predictions $v_{1\cdots n}$ are given to the control agent as input features with which an action a_{t+1} is chosen according to π_{control}. After taking an action, the agent observes a resulting reward R_{t+1} and a following observation o_{t+1} and the cycle repeats.

Three parameters—the discount γ, policy π and the cumulant c determine what aspect of the environment each prediction is *about*. The cumulant determines the signal of interest from the environment, and the policy defines what the agent is doing during the prediction. We define meta-weights w_π and w_c that determine the cumulant and policy a prediction is conditioned on. These meta-parameters are incrementally learned alongside the GVF they specify, affecting the GVF by determining how the values should change by modifying w_{gvf} during each temporal difference update.



(a) A depiction of the agent-environment relationship showing how the agent processes information from the environment, and chooses an action.

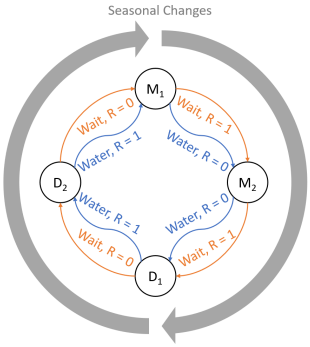


(b) A depiction of the indirect relationship between the error and the underlying weights.

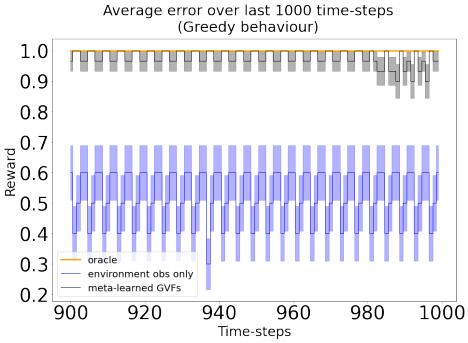
Figure 1: Relationship between the sub-components of the agent and its environment. Denoted in green is the environments feedback in the form of reward/TD error.

Algorithm 1 A meta-gradient approach to self-supervise prediction selection to inform control.

INITALISE: set control agent weights w_{control}, GVF weights $w_{\text{gvf},i\cdots n}$, and meta weights $w_{c,i}$, $w_{\pi,i}$. Choose activations ϕ_{cumulant} and ϕ_{policy}. Choose step-size α for the control agent, GVFs, and meta-parameters independently. Set an L2 λ. Set an ϵ e-greedy for action-selection.
START: Make initial observation o_0, take initial action a_0. The gvf-state$_0$ is o_0, a_0.
Produce GVF estimates V_0; these estimates form control-state$_0$.
for t=1, to final time-step T:
With probability ϵ select a random action a_t.
Else select action $a_t = argmax_a Q(\phi(v_t), a_t)$.
Observe o_{t+1} and r_{t+1} resulting from action a_t.
Perform meta-gradient descent
Take gradient steps on $\mathcal{L}$ with respect to both w_{policy} and w_{cumulant}.
Update GVFs
Output current cumulant $c = \phi_{\text{cumulant}}(o_t, w_{\text{cumulant}})$.
Output current policy $\pi_{\text{gvf}} = \phi_{\text{policy}}(o_t, w_{\text{policy}})$.
Approximate state for GVFs: gvf-state $= o_t, a_t, v_t$.
Update each GVF's parameters w_{gvf} given their computed π, c, and fixed γ.
Update control policy
Output current estimate for each GVF given v_{t+1}(gvf-state$_t, w_{\text{gvf}}$).
Approximate state for control learner: control-state$_{t+1} = v_{t+1}$.
Update control agent's Q values given control-state$_t$, a_t, r_{t+1},
And control-state$_{t+1}$.
end



(a) There are four states: 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 predictions that are known to express the seasons (in orange), 3) two additional predictions that are updated using meta-gradient descent (in black).

Figure 2: Monsoon World and comparison of learned policies. Each independent agent in 2b is averaged over 30 independent trials. Error bars are standard error.

There is an indirect relationship between the updates to the meta-weights, and the agent's TD error δ_{control}; through this relationship we can express a gradient that describes how the underlying weights w_{pi} and w_c which shape the meaning of a particular GVF influence the agent's Temporal Difference error: $\frac{\partial \delta_{\text{control}}}{w_\pi} = \frac{\partial \delta_{\text{control}}}{\partial w_{\text{control}}} \frac{\partial w_{\text{control}}}{\partial w_{\text{gvf}}} \frac{\partial w_{\text{gvf}}}{\partial w_\pi}$. By this relationship, we can construct a loss function $\mathcal{L}_\pi(w_{\text{control}}, w_\pi) = \delta^2_{\text{control}}$ for the policy and the cumulant. Using meta-gradient descent, the underlying parameters w_c and w_π that output the cumulant c and policy π that a given agent is following perform meta-gradient descent can be updated as so:$w_\pi \leftarrow w_\pi - \alpha_\pi \nabla_{w_\pi} \mathcal{L}_\pi$ and $w_c \leftarrow w_c - \alpha_c \nabla_{w_c} \mathcal{L}_c$.

3 Monsoon World: A Partially Observable Environment

We evaluate meta-gradient discovery of GVFs using a partially observable control problem, Monsoon World (Figure 2a). In Monsoon World, there are two seasons: monsoon and drought. 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.

In this environment the agent cannot observe the underlying season that determines the outcome of their action. While the agent cannot directly observe seasons, it can observe something impacted by the seasons–the result of a given action.

This monsoon problem can be solved, and an optimal policy found, if the agent reliably estimates how long until watering produces a particular result. This can be done by learning *echo GVFs* (Schlegel et al., 2021). Echo GVFs estimate the time to an event using a state-conditioned discount and cumulant. In this case, estimating how long until there is growth $o_{i,t} = 1$, or there is no growth $o_{i,t} = 0$ when watering: $c = 1 \quad \text{if} \quad o_{i,t} = 1$; else $\quad c = 0$. Similarly, a state-dependent discounting function terminates the accumulation $\gamma(s_t, a_t, s_{t+1})$, where $\gamma = 0 \quad \text{if} \quad c = 1$; else $\quad 0.9$. These estimates can be learnt off-policy using a deterministic policy (e.g., "if the agent waters" $\pi = [0, 1]$).

Having constructed the aforementioned GVFs, we are now able to express what is hidden from our observation stream: how long until the next season. While no information was given about the season, by relating what is sensed by the agent with the actions that were taken by the agent, we are able to learn about the seasons indirectly.

4 Discovering GVFs in Monsoon World

We now answer the question: "Can an agent find useful predictions by performing meta-gradient descent?" To do so, we compare three different agent configurations (Figure 2b): 1) a baseline agent that only receives environmental observations as inputs, 2) an agent that in addition to the environmental observations, receives the estimates of two GVFs with cumulants and policies known to be effective in capturing the underlying seasons, and 3) an agent that has two additional predictions that are learned through meta-gradient descent.

When GVFs are specified via meta-gradient descent, we initialise policies to an equiprobable weighting of actions and cumulants to an equal weighting of observations. The policy weights are passed through a Softmax activation function so that their sum is between 1 and 0, and the cumulants are passed through a sigmoid activation to bound the cumulant between [0,1]. The meta-weights are updated each time-step incrementally. We apply L2 to the loss with $\lambda = 0.001$. Additional details are in Appendix A.

As introduced in Section 3, observations alone are insufficient to determine the optimal action on a given time-step. 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). When expertly specified estimates are learned and provided as inputs in addition to the environmental observations (orange). the learned policy is approximately optimal: using predictions that estimate the time to each season optimal actions are taken most of the time. By 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.*

5 Limitations & Future Work

We introduced a new approach to meta-learning predictions where GVFs outputs are used as features by a control agent. We found that an agent with no prior knowledge of the environment was able to select predictions that yielded performance equitable to agents using expertly chosen predictive features. This success opens up several interesting questions for future work: 1) How well does meta-gradient selection perform in domains with higher-dimensional observations and more actions? 2) How well would meta-gradient GVF selection perform in non-stationary domains? Finally, our proposed approach is sensitive to the meta step-sizes that govern the incremental updates. How optimisers or step-size adaptations could improve robustness has yet to be explored.

6 Conclusion

In this paper we demonstrated how predictions in the form of GVFs can be decided upon and learned by meta-gradient descent alongside a policy for agent action selection. Doing so, we enable our agent to learn about its environment in a self-supervised and independent way. We evaluate our approach on a partially-observable MDP, called Monsoon world. Our results demonstrate that an agent can independently specify GVFs that enable performance comparable to expertly chosen predictions that remove the partial-observability. This work therefore tackles one of the most important problems in prediction-based self-supervised learning.

References

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

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

Gilbert, D. (2009). *Stumbling on happiness*. Vintage Canada.

Günther, J., Pilarski, P. M., Helfrich, G., Shen, H., and Diepold, K. (2016). Intelligent laser welding through representation, prediction, and control learning: An architecture with deep neural networks and reinforcement learning. *Mechatronics*, 34:1–11. System-Integrated Intelligence: New Challenges for Product and Production Engineering.

Kearney, A., Koop, A., and Pilarski, P. M. (2021). What's a good prediction? Issues in evaluating general value functions through error.

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

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

Schlegel, M., Jacobsen, A., Abbas, Z., Patterson, A., White, A., and White, M. (2021). General value function networks. *Journal of Artificial Intelligence Research*, 70:497–543.

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

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

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., and Precup, D. (2011). 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.

Veeriah, V., Hessel, M., Xu, Z., Lewis, R. L., Rajendran, J., Oh, J., van Hasselt, H., Silver, D., and Singh, S. (2019). Discovery of useful questions as auxiliary tasks. *CoRR*, abs/1909.04607.

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

A Experiment Details

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

A.1 Function Approximators

We use different function approximators to transform the given inputs to an *agent-state* $s_t = \phi(o_t, v_t)$. Echo GVFs are in log-space; before using them as inputs, we apply a transformation to them as follows:

Algorithm 2 Log-transform of prediction estimates

```
# Where v is are the value estimate from n GVFs.
transform(v) :
    v ← clip(log(v)/log(0.9), 0, 1)
    return v
```

We use state aggregation to transform the estimates produced by each GVF into a binary feature vector s_t such that the value.

Algorithm 3 State aggregation of predictions

```
# Where v is are the value estimate from n GVFs.
# Where memsize is the allocated length for the binary feature vector.
state(v, memsize) :
    s = zeros(memsize)
    i ← v[0] + v[1] * 10    # this assumes that each v_i < 10
    s[i] = 1
    return s
```

The function approximation for each agent is as follows:

Environment Observations Only

1. Control Agent: state aggregation.
2. GVFs: state aggregation.
3. Meta-parameters: n/a.

Expert Chosen Predictions & Environment Observations

1. Control Agent: state aggregation.
2. GVFs: state aggregation.
3. Meta-parameters: n/a.

Meta-gradient Learned Predictions & Environment Observations

1. Control Agent: no function approximator; a linear combination of weights and inputs.
2. GVFs: state aggregation.
3. Meta-parameters: no function approximator; a linear combination of weights and inputs.

A.2 Parameter Settings

Parameters were chosen by performing a sweep across different values, choosing the best performing combination for each agent.

Agent configuration	ϵ	$\alpha_{control}$	α_{gvfs}	α_{π}	α_{c}
Environment Obs Only	0.1	0.01	0.1	n/a	n/a
Expert Chosen Predictions	0.1	0.01	0.1	n/a	n/a
Meta-gradient Learned Predictions	0.5	0.0001	0.1	0.001	0.1

Table 1: Parameter settings for different agent configurations

A.3 Meta-parameter Specification

The policy π is a deterministic policy. The meta-weights determine the policy a GVF is conditioned on, but they are not a function of the observations: $\pi \leftarrow \text{softmax}(w_{\pi})$.

The cumulant c is a function of the observations such that $\text{sigmoid}(w_c^{\top} o_t)$, where w_c are the meta-weights for the cumulant, and o_t is the present environment observation.