World Value Functions: Knowledge Representation for Learning and Planning

Geraud Nangue Tasse*, Steven James and Benjamin Rosman

University of the Witwatersrand, Johannesburg, South Africa

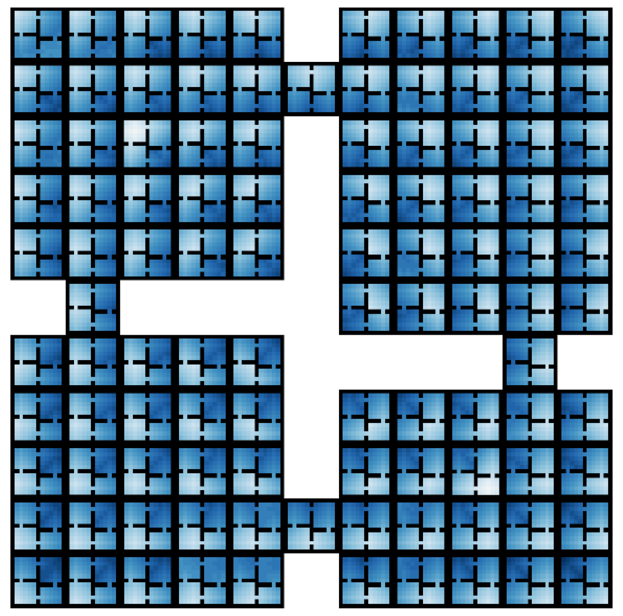
WITS

RAIL LAB

A general value function with mastery of the world (provably) that encodes the solution to the current task and has downstream zero-shot abilities.

Introduction

- How do we learn and represent knowledge that is **sufficient** for a **general agent** that needs to **solve multiple tasks** and **plan** in a given world?



- General value functions (GVFs) [1] are a general approach that tries to answer this question. Consider for example a 4-rooms gridworld. A GVF here can be defined by defining a set of goals $G \coloneqq S$ and a goal-specific reward function $\boldsymbol{R}(s, g, a, s') \coloneqq$ ***1 if s==g else 0.*** *The GVF is given by,*

$$\boldsymbol{Q}(s,g,a) = \mathbb{E}_s\left[\boldsymbol{R}(s,g,a,s_1) + \sum_{t=1}^{\infty} \gamma^t \boldsymbol{R}(s_t,g,a_t,s_{t+1})\right]$$

- GVFs can also be learned efficiently in non-tabular settings using universal value function approximators (UVFAs) [2].
- However, what is the **origin of goals** and how to define **goal-specific rewards** in general? WVFs are a subset of GVFs that answer these questions—goals are simply **states with terminal transitions**, while goal rewards are simply task rewards with a **penalty term** added for achieving wrong goals.

[1] Sutton, Richard S., et al. **Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction**. In ICML 2011.
[2] Schaul, Tom, et al. **Universal value function approximators**. ICML 2015.
[3] Sutton, Richard S. **Integrated architectures for learning, planning, and reacting based on approximating dynamic programming**.MLP 1990.

PAPER

World Value Functions

- We first define the agent's **internal goals** $\mathcal{G}$ as all states with terminal transitions.
- The WVF $\boldsymbol{Q}(s,g,a)$ for a task in a given world is defined by the agent's pseudo-reward function:

$$\boldsymbol{R}(s,g,a,s') = \begin{cases} R_{MIN} & \textit{if } g \neq s \textit{ and } s' \textit{is terminal,} \\ R(s,a,s') & \textit{otherwise} \end{cases}$$

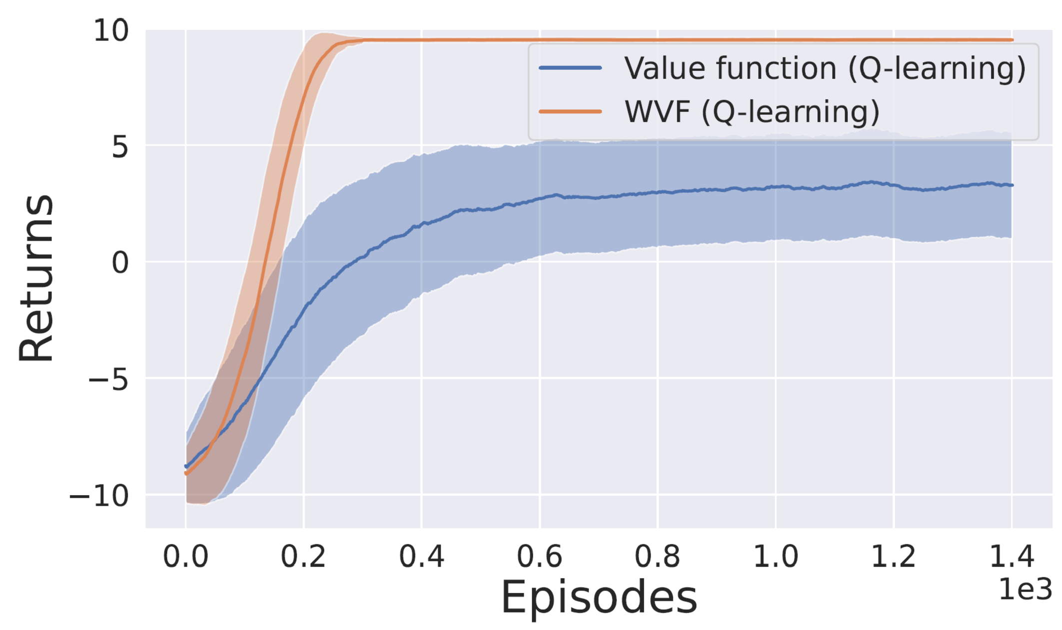
where $\mathrm{R_{MIN}}$ is a **large penalty** the agent gives itself for achieving the wrong internal goals.
- This leads to **mastery** (provably): The agent learns how to achieve all internal goals.
- The regular task rewards, value function, and policy can always be recovered (provably):

$$R(s,a,s') = \max_g \boldsymbol{R}(s,g,a,s'),\ Q(s,a) = \max_g \boldsymbol{Q}(s,g,a)$$
$$\pi(s) \sim argmax_a \max_g \boldsymbol{Q}(s,g,a)$$

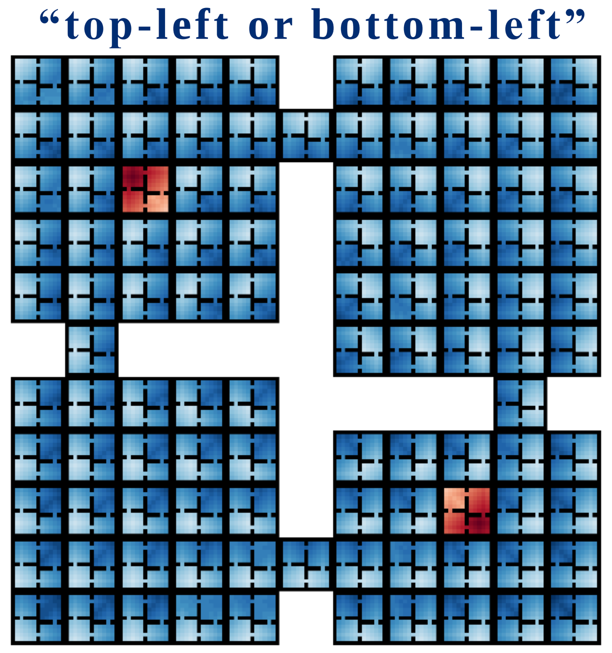
- Finally, WVFs encode the dynamics of the world. When $\mathcal{G} = \mathcal{S}$, $p(.|s,a)$ can be estimated by solving the system of Bellman equations:

$$\boldsymbol{Q}^*(s,g,a) = \sum_{s' \in S} p(s'|s,a)[\boldsymbol{R}(s,g,a,s') + \boldsymbol{V}^*(s,g)]$$

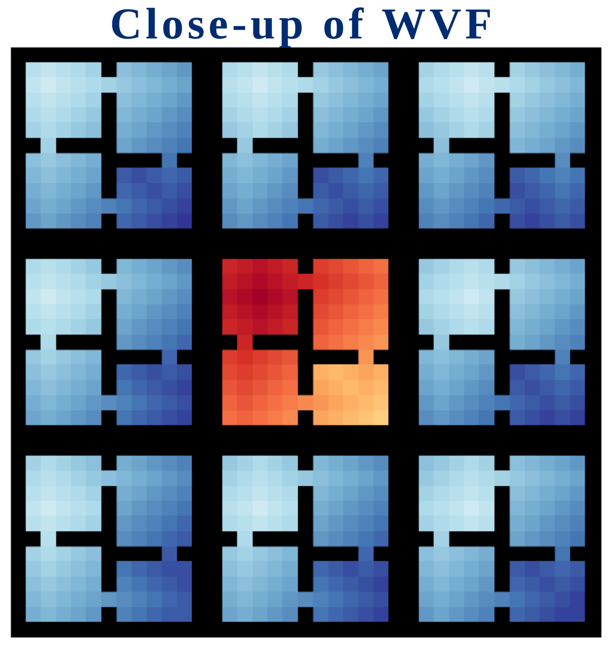
$\forall g \in \mathcal{G}$. This can then be used for model-based RL



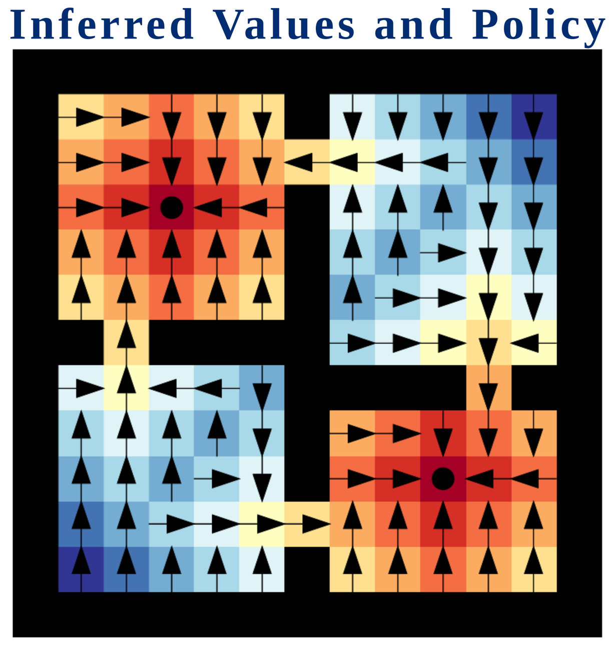
Learned WVF for the task "top-left or bottom-left"



Close-up of WVF



Inferred Values and Policy

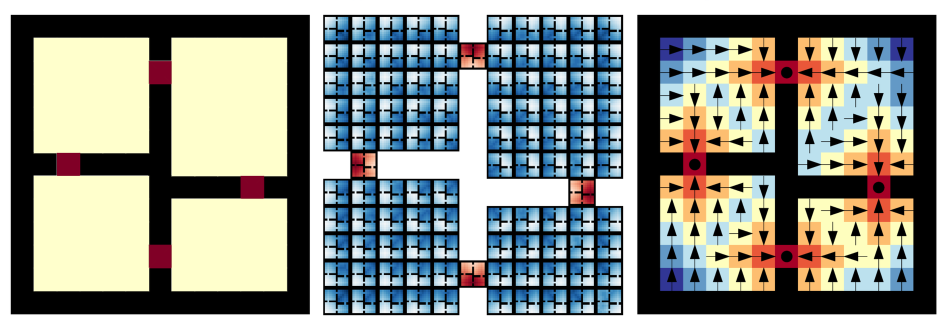


Zero-shot Values and Policies from Rewards

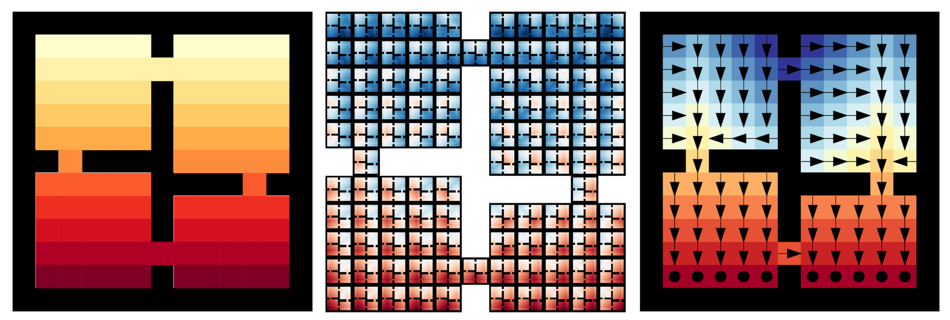
- We can obtain the WVF $\boldsymbol{Q}^*_M$ for **any** task given its goal rewards R_G and an **arbitrary** WVF $\boldsymbol{Q}^*$:

$$\boldsymbol{Q}^*_M(s,g,a) \approx \boldsymbol{Q}^*(s,g,a) + [\max_a R_G(g,a) - \max_a \boldsymbol{Q}^*(g,g,a)]$$

Navigate to a hallway



Navigate to the bottom

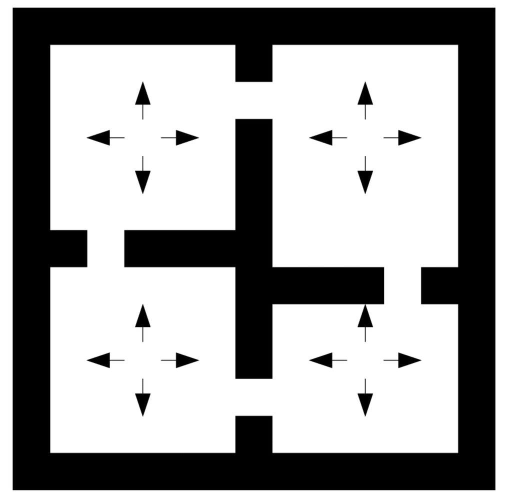


Fast RL with Zero-shot Dynamics

Algorithm 2: Dyna for WVFs using inferred transition functions

```
Initialise: WVF Q̄, Reward function R, goal buffer
            𝒢, learning rate α
foreach episode do
    Observe initial state s ∈ 𝒮 and sample g ∈ 𝒢
    while episode is not done do
        a ← { arg max_{a∈𝒜} Q̄(s, g, a)   w.p. 1 − ε
            { a random action            w.p. ε
        Execute a, observe reward r and next state s′
        R(s, a, .) ← r
        if s′ is absorbing then 𝒢 ← 𝒢 ∪ {s}
        for g′ ∈ 𝒢 do
            r̄ ← R̄_MIN if g′ ≠ s and s ∈ 𝒢 else r
            δ ← [r̄ + max_{a′} Q̄(s′, g′, a′)] − Q̄(s, g′, a)
            Q̄(s, g′, a) ← Q̄(s, g′, a) + αδ
        repeat N times
            s ← random previous state
            a ← random previous action taken in s
            r ← R(s, a, .)
            s′ ← Solving 𝒩(s) Bellman equations
            MSE ← 1/|𝒩(s)| Σ_{g∈𝒩(s)} (Q̄(s, g, a)−
                      [R̄(s, g, a, s′) + V̄(s′, g)])
            if MSE ≤ threshold then
                for g′ ∈ 𝒢 do
                    r̄ ← R̄_MIN if g′ ≠ s and s ∈ 𝒢
                      else r
                    δ ← [r̄ + max_{a′} Q̄(s′, g′, a′)] −
                      Q̄(s, g′, a)
                    Q̄(s, g′, a) ← Q̄(s, g′, a) + αδ
        s ← s′
```

Inferred Transitions



Imagined Rollouts