Towards Sample-Efficient Multi-Objective Reinforcement Learning

Lucas N. Alegre

INF - Universidade Federal do Rio Grande do Sul
AI-Lab - Vrije Universiteit Brussel

.inf INSTITUTO DE INFORMÁTICA UFRGS

VUB

ARTIFICIAL INTELLIGENCE RESEARCH GROUP

About Me

- Bs.C. Computer Science *cum laude* at INF-UFRGS (2016 - 2020)

About Me

- Bs.C. Computer Science *cum laude* at INF-UFRGS (2016 - 2020)

- Ph.D. on Multi-Objective Reinforcement Learning (2021 - ongoing)
  - Advisor: Prof. Ana Bazzan (INF-UFRGS)
  - Co-Advisor: Prof. Bruno da Silva (UMASS)

About Me

- Bs.C. Computer Science *cum laude* at INF-UFRGS (2016 - 2020)

- Ph.D. on Multi-Objective Reinforcement Learning (2021 - ongoing)
  - Advisor: Prof. Ana Bazzan (INF-UFRGS)
  - Co-Advisor: Prof. Bruno da Silva (UMASS)


- Ph.D. *sandwich period* at AI-Lab VUB (Aug 2022 – Jul 2023)
  - Advisor: Prof. Ann Nowé

Reinforcement Learning

Agent

Environment

Action A_t

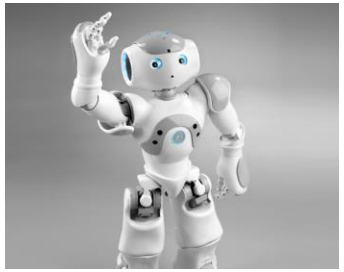
Next State S_{t+1}

Reward R_t

Model-Free RL

- Markov Decision Process (MDP) $M = (S, A, {\color{blue}p}, {\color{red}R})$

Agent



$A_t \sim \pi(\cdot \mid S_t)$

Environment

$R_t \sim {\color{red}R}(S_t, A_t, S_{t+1})$

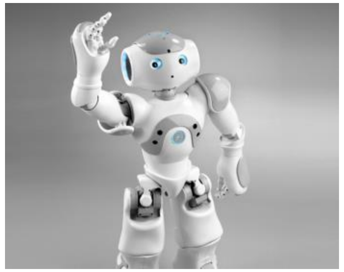
$S_{t+1} \sim {\color{blue}p}(\cdot \mid S_t, A_t)$

- Learn a policy π that maximizes $\mathbb{E}\left[\,\sum_t \gamma^t R_t \mid \pi, {\color{red}R}, {\color{blue}p}\right]$

Model-Free RL

- Markov Decision Process (MDP) $M = (S, A, \color{blue}{p}, \color{red}{R})$

Agent



$A_t \sim \pi(\cdot \mid S_t)$

Environment

$R_t \sim \color{red}{R}\color{black}(S_t, A_t, S_{t+1})$
$S_{t+1} \sim \color{blue}{p}\color{black}(\cdot \mid S_t, A_t)$

In model-free RL,
we <u>do not know</u>
$\color{blue}{p}$ and $\color{red}{R}$

- Learn a policy π that maximizes $\mathbb{E}\left[\sum_t \gamma^t R_t \mid \pi, \color{red}{R}, \color{blue}{p}\right]$

Value-Based Model-Free RL

- Action-Value function

$$q^{\pi}(s,a) = \sum_{s'} p(s'|s,a)\,[R(s,a,s') + \gamma q^{\pi}(s',\pi(s'))]$$

Value-Based Model-Free RL

- Action-Value function

$$q^{\pi}(s,a) = \sum_{s'} p(s'|s,a)\,[R(s,a,s') + \gamma q^{\pi}(s',\pi(s'))]$$

- Q-learning

$$Q(s,a) \;\coloneqq\; Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$$

r and s' are sampled from the environment real dynamics R and p

Value-Based Model-Free RL

- Action-Value function

$$q^{\pi}(s,a) = \sum_{s'} p(s'|s,a)\,[R(s,a,s') + \gamma q^{\pi}(s',\pi(s'))]$$

- Q-learning

$$Q(s,a) \coloneqq Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$$

r and s' are sampled from the environment real dynamics R and p

Multi-Objective Reinforcement Learning

Agent

Environment

Action A_t

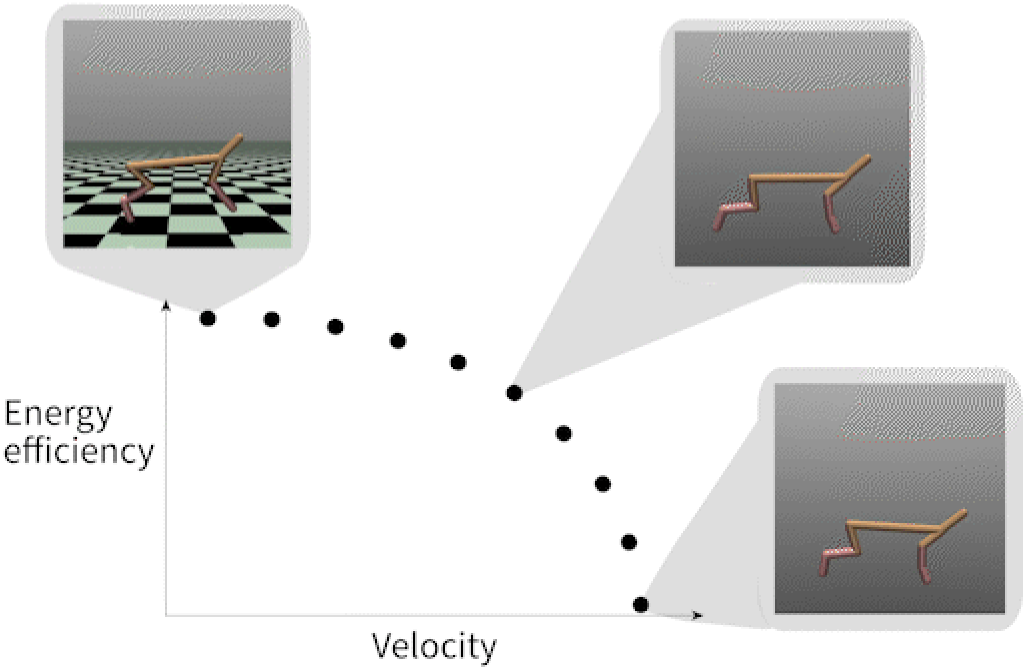
Next State S_{t+1}

Reward vector

$\boldsymbol{R}_t = [R_1 \dots R_\mathrm{m}]$

Multi-Objective Reinforcement Learning

Goal: Learn a set of policies $\Pi = \{\pi_1, \ldots, \pi_n\}$ guaranteed to contain an optimal policy for ***any*** preferences $\boldsymbol{w}$ over objectives



$$v_{\boldsymbol{w}}^{\pi} = \boldsymbol{v}^{\pi} \cdot \boldsymbol{w}$$

$$= v_1^{\pi} w_1 + \cdots + v_m^{\pi} w_m$$

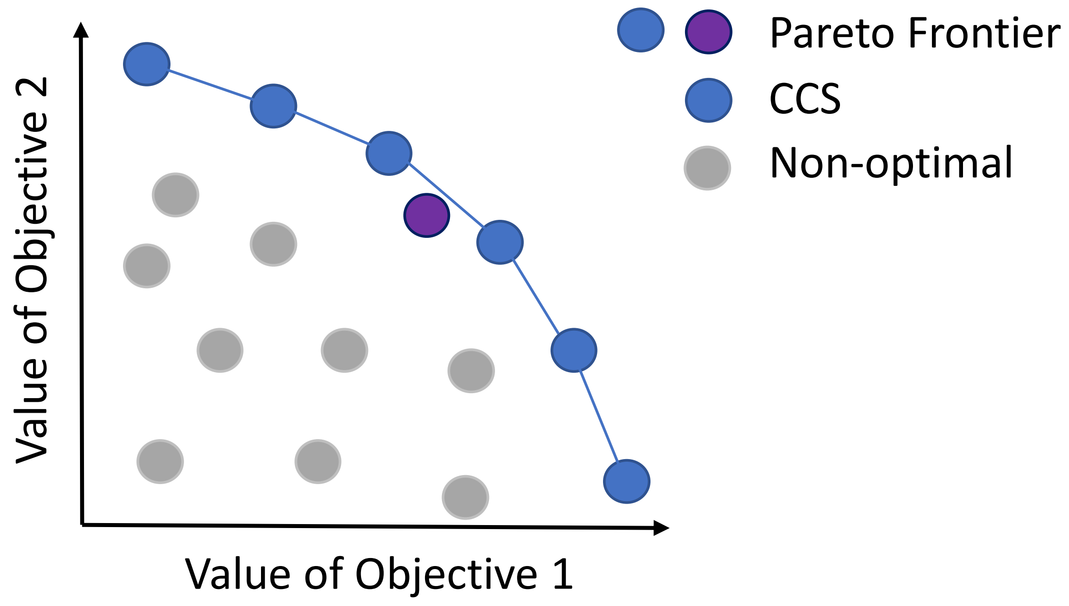
Value w.r.t. m-th objective

Multi-Objective Reinforcement Learning

Convex Coverage Set (CCS)

Optimal solution when the preferences are linear:

$$\mathrm{CCS} \equiv \{ \mathbf{v}^{\pi} \in \mathcal{F} \mid \exists \mathbf{w} \text{ s.t. } \forall \mathbf{v}^{\pi'} \in \mathcal{F}, \mathbf{v}^{\pi} \cdot \mathbf{w} \geq \mathbf{v}^{\pi'} \cdot \mathbf{w} \}$$



Optimal policy w.r.t. *any* convex combination of rewards is in the **CCS**!

Sample-Efficient Multi-Objective Learning via Generalized Policy Improvement Prioritization

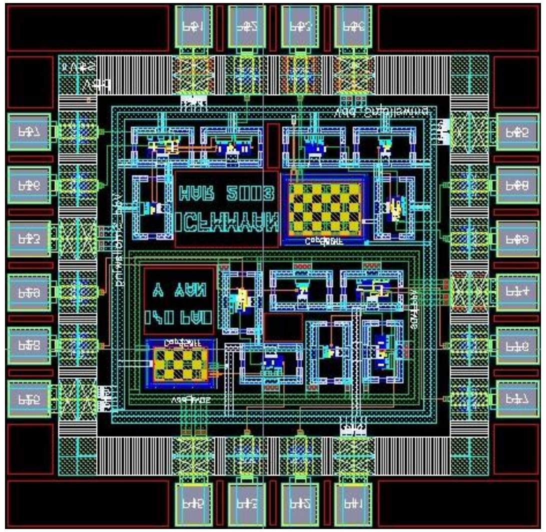
Lucas N. Alegre[1,2] Ana L. C. Bazzan[1] Diederik M. Roijers[2]
Ann Nowé[2] Bruno C. da Silva[3]

[1] Universidade Federal do Rio Grande do Sul [2] Vrije Universiteit Brussel [3] University of Massachusetts
lnalegre@inf.ufrgs.br

RL Success Cases

Mastering the game of Go with deep neural networks and tree search. Silver, D., Huang, A., Maddison, C. *et al.*. *Nature* **529**, (2016).

Autonomous navigation of stratospheric balloons using reinforcement learning. Bellemare, M.G., Candido, S., Castro, P.S. *et al.*. *Nature* **588**, (2020)



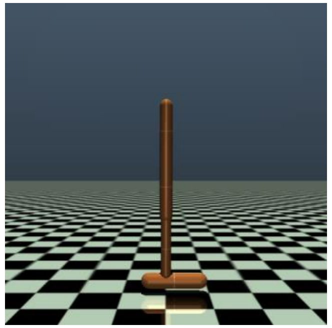
A graph placement methodology for fast chip design. Mirhoseini, A., Goldie, A., Yazgan, M. *et al.* . *Nature* **594**,(2021).

… and many other applications

Sample Efficiency in RL

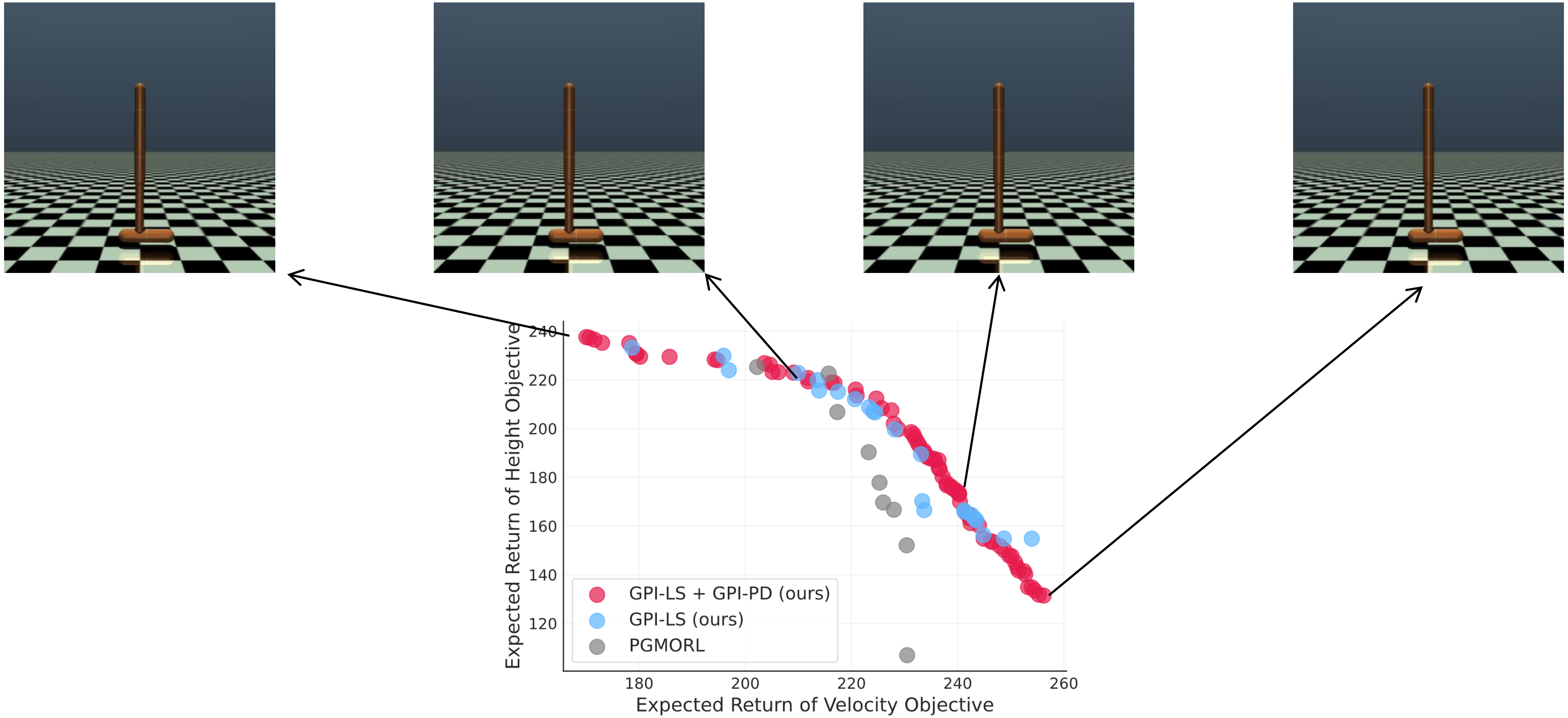
- Thousands of environment interactions are required to learn

one policy



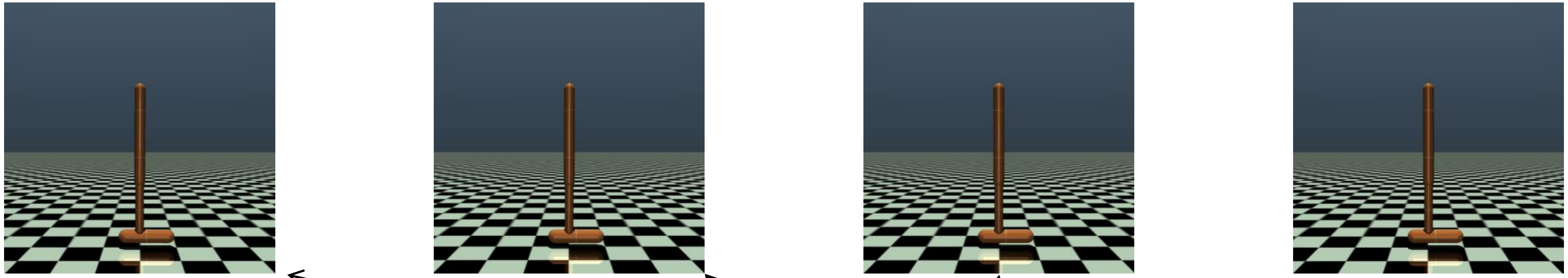
Sample Efficiency in MORL

- Many thousands of environment interactions are required to learn **a set** of policies! (one for each user preference)



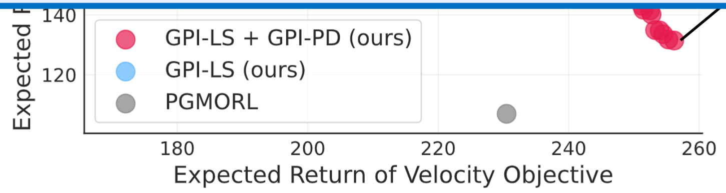
Sample Efficiency in MORL

- Many thousands of environment interactions are required to learn **a set** of policies! (one for each user preference)



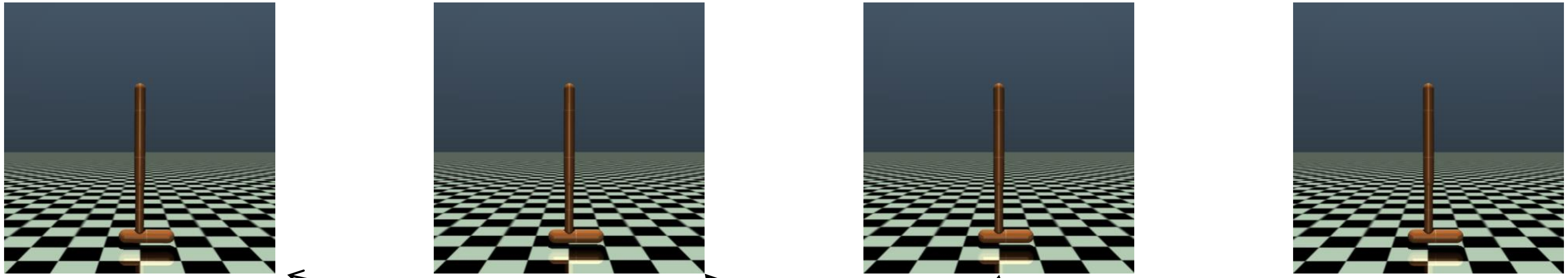
Infeasible in real-world settings!

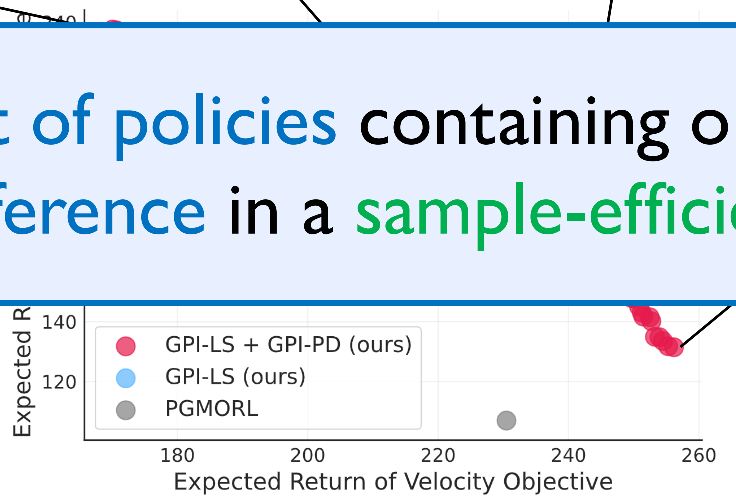
e.g. autonomous driving, healthcare, robotics



Sample Efficiency in MORL

- Many thousands of environment interactions are required to learn **a set** of policies! (one for each user preference)





How to learn a set of policies containing optimal policies for *any* user preference in a sample-efficient manner?

Main Contributions

Two Generalized Policy Improvement (GPI)-based prioritization schemes that improve sample-efficiency in MORL:

GPI Linear Support (GPI-LS)

- Identify the most promising preferences/objectives to train on
- Guaranteed identification of optimal (or ϵ-optimal) sets of policies

GPI-Prioritized Dyna (GPI-PD)

- Identify relevant previous experiences when learning a new policy
- First model-based MORL method for continuous states/actions

GPE & GPI

Generalized Policy Evaluation (GPE)

is the computation of the value function of a policy π

on a ***set*** *of tasks (reward functions)*

Barreto, A. et al. *Fast reinforcement learning with generalized policy updates*. Proceedings of the National Academy of Sciences, 2020.

GPE & GPI

Generalized Policy Evaluation (GPE)

is the computation of the value function of a policy π

on a ***set*** *of tasks (reward functions)*

Good news:

In MORL under linear preferences, we can

perform GPE *without having to test/deploy the policy*

$$q_w^{\pi}(\mathrm{s},\mathrm{a}) = \mathbf{q}^{\pi}(\mathrm{s},\mathrm{a}) \cdot w \quad \textit{for any } w \in \mathcal{W}$$

Barreto, A. et al. *Fast reinforcement learning with generalized policy updates*. Proceedings of the National Academy of Sciences, 2020.
Alegre, L. N. et al. *Optimistic linear support and successor features as a basis for optimal policy transfer*. ICML, 2022.

Generalized Policy Improvement (GPI)

Generalized Policy Improvement (GPI)

is the computation of a policy π' that improves

over a ***set*** *of policies* $\pi \in \Pi$ *given* ***any*** *new reward weights* $\boldsymbol{w}$

$$\boxed{\pi^{GPI}(s\,;w) = \arg\max_{a \in \mathcal{A}} \max_{\pi \in \Pi} q_w^{\pi}(s,a)}$$

[1] *Successor Features for Transfer in Reinforcement Learning*. Barreto et al. (NIPS 2017)

Generalized Policy Improvement (GPI)

Generalized Policy Improvement (GPI)

is the computation of a policy π' that improves

over a ***set*** *of policies* $\pi \in \Pi$ *given* ***any*** *new reward weights* $\boldsymbol{w}$

$$\boxed{\pi^{GPI}(s\,;w) = \arg\max_{a \in \mathcal{A}} \max_{\pi \in \Pi} q_w^{\pi}(s,a)}$$

GPI Theorem[1]: $q_w^{GPI}(s,a) \geq \max_{\pi \in \Pi} q_w^{\pi}(s,a)$ *for any* $w \in \mathcal{W}$

[1] *Successor Features for Transfer in Reinforcement Learning*. Barreto et al. (NIPS 2017)

GPI Linear Support

- Iteratively learns a policy set Π whose values approximte the CCS

GPI Linear Support

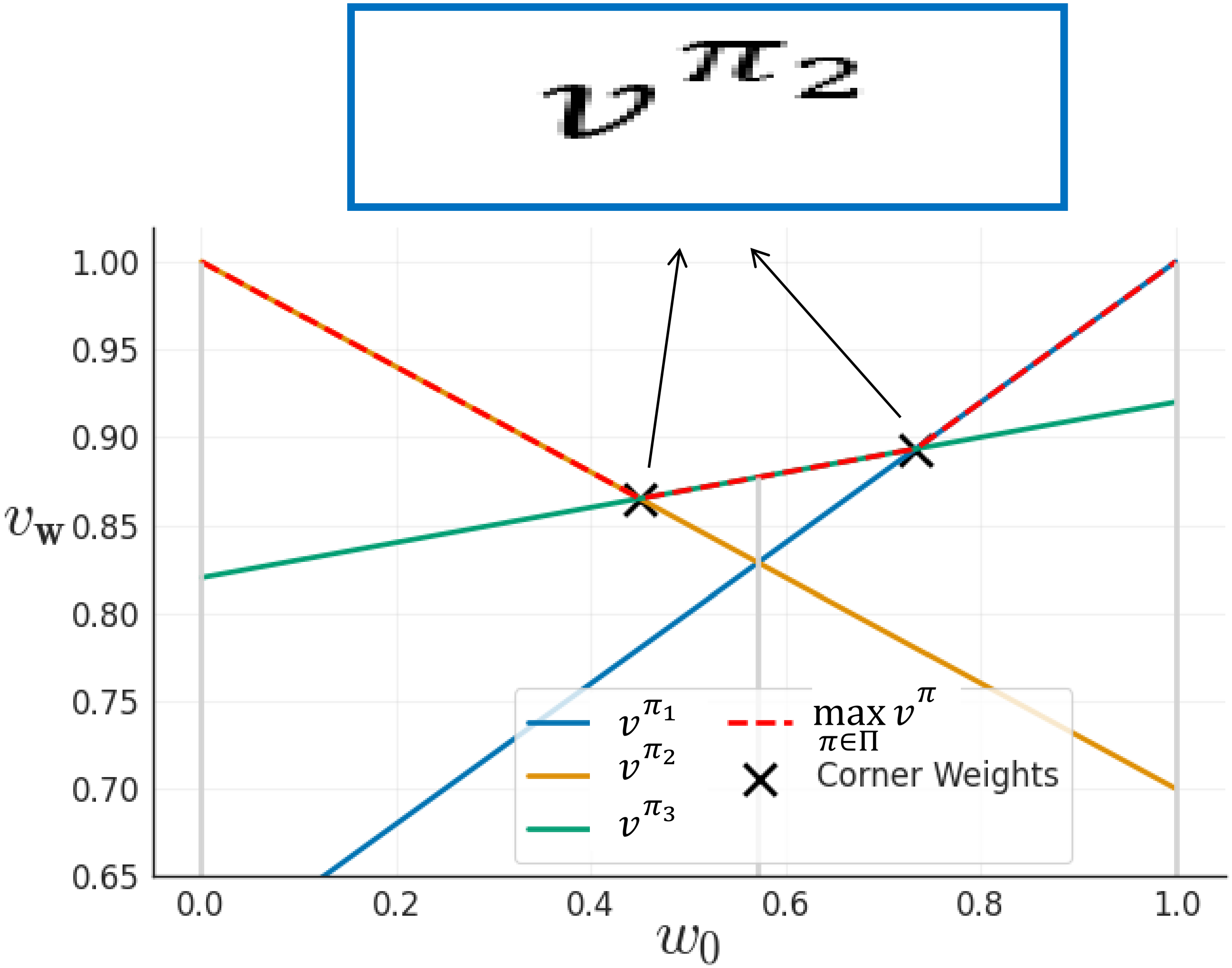
- Iteratively learns a policy set Π whose values approximte the CCS

Key idea: GPI Prioritization

- Identifying the most promising preferences to train on
  $\mapsto$ focus on corner weights
- Prioritize reward weights w.r.t. performance improvement given by GPI:

$$\boxed{\arg\max_{\mathbf{w}\in\mathcal{W}_{\text{corner}}} \left(v_{\mathbf{w}}^{\text{GPI}} - \max_{\pi\in\Pi} v_{\mathbf{w}}^{\pi}\right)}$$

GPI Linear Support



- Maximum improvement is guaranteed to be in one of the corner weights (Thm. 3.2)
- Iteratively:
  - Selects the corner weight with *higher* GPI priority
  - Learns an improved policy for the selected reward weights

GPI Linear Support

Algorithm 1: GPI Linear Support (GPI-LS)

THEOREM 3.3. *Let* NewPolicy$(\mathbf{w}, \Pi)$ *in Alg. 1 be any algorithm that returns an optimal policy,* $\pi^*_{\mathbf{w}}$*, for a given weight vector* $\mathbf{w}$*. Then, Alg. 1 is guaranteed to find a* CCS *in a finite number of iterations.*

if $\mathcal{W}_{\text{corner}}$ *is empty* **then**

THEOREM 3.5. *Let* NewPolicy$(\mathbf{w}, \Pi)$ *in Alg. 1 be an algorithm that produces an* ϵ*-optimal policy,* $\pi_{\mathbf{w}}$*, when its termination condition is met (when it returns* done $=$ True*); that is,* $v^*_{\mathbf{w}} - v^{\pi_{\mathbf{w}}}_{\mathbf{w}} \leq \epsilon$*. Then, Alg. 1 is guaranteed to return an* ϵ*-*CCS*.*

$\Pi, \mathcal{V} \leftarrow \text{RemoveDominated}(\Pi, \mathcal{V})$

Value-Based Model-Free RL

- Action-Value function

$$q^{\pi}(s,a) = \sum_{s'} p(s'|s,a)\,[R(s,a,s') + \gamma q^{\pi}(s',\pi(s'))]$$

- Q-learning

$$Q(s,a) \;\coloneqq\; Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right)$$

r and s' are sampled from the environment real dynamics R and p

What if we have access/learn R and p ?

Model-Based RL

- Learns a model p of the environment

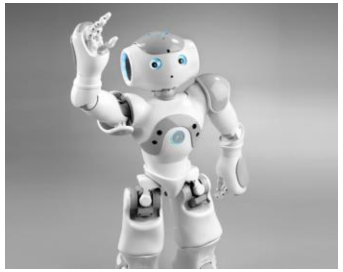
$$(s', r) \sim p(\cdot \mid s, a)$$

Model-Based RL

- Learns a model p of the environment

$$(s', r) \sim p(\cdot \mid s, a)$$

Agent



$$A_t \sim \pi(\cdot \mid S_t)$$

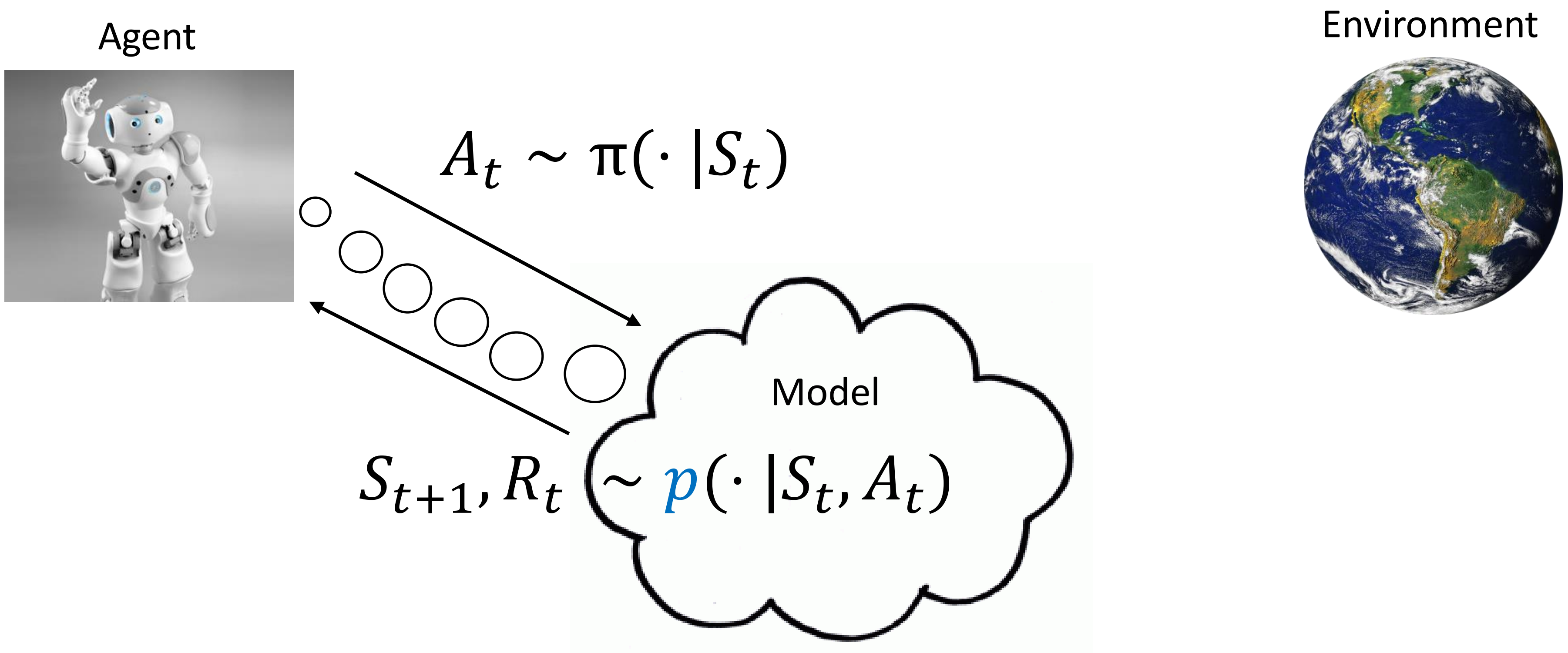
Environment

$$R_t \sim R(S_t, A_t, S_{t+1})$$
$$S_{t+1} \sim p(\cdot \mid S_t, A_t)$$

Model-Based RL

- Learns a model p of the environment

$$(s', r) \sim p(\cdot\,|s, a)$$



Model-Based MORL

- Increase sample-efficiency in RL using a learned model of the environment

Model-Based MORL

- Increase sample-efficiency in RL using a learned model of the environment
- Few model-based methods have been explored in MORL
- We learn a **model** that predicts the **next state** and **reward vector**:

$$p_\varphi(S_{t+1}, \mathbf{R}_t | S_t, A_t)$$

This model can be used to learn policies

for *<u>any</u>* given preferences!

GPI Prioritized Dyna (GPI-PD)

Policies learned via a Dyna-style approach

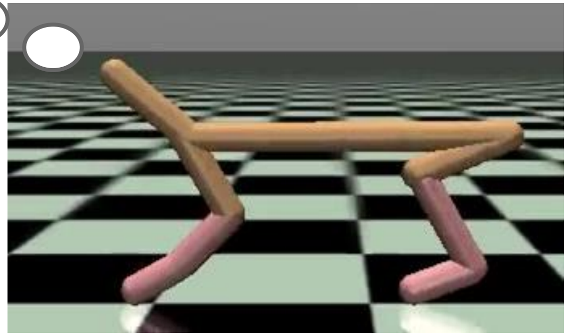
for *H Dyna steps* **do** ▷ GPI-Prioritized Dyna

Sample $S \sim \mathcal{B}$ according to $P_{\mathbf{w}_t}$ (Eq. (10))

$A \leftarrow \pi^{\text{GPI}}(S; \mathbf{w}_t)$; $(\hat{S}', \hat{\mathbf{R}}) \sim p_{\varphi}(\cdot|S, A)$

Add $(S, A, \hat{\mathbf{R}}, \hat{S}')$ to $\mathcal{B}_{\text{model}}$

$$\boxed{P_{\mathbf{w}}(s,a) \propto q^{\text{GPI}}_{\mathbf{w}}(s,a) - q^{\pi}_{\mathbf{w}}(s,a)}$$



Prioritizes experiences for which GPI results in larger performance improvements

GPI-PD with Function Approximation

- Conditioned Action-Value Functions

$$\mathbf{Q}_\theta(s, a, \mathbf{w}) \approx \mathbf{q}^{\pi_{\mathbf{w}}}(s, a)$$

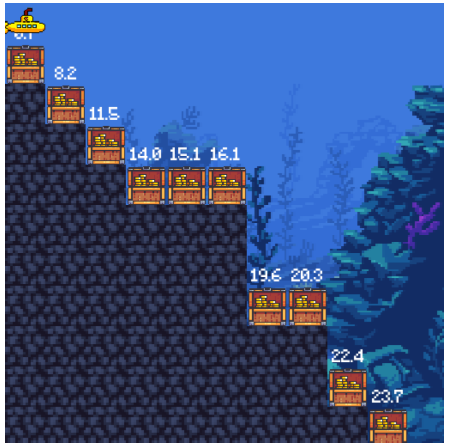
$$\pi^{\mathrm{GPI}}(s; \mathbf{w}) \in \arg\max_{a \in \mathcal{A}} \max_{\mathbf{w}' \in \mathcal{M}} \mathbf{Q}_\theta(s, a, \mathbf{w}') \cdot \mathbf{w}$$

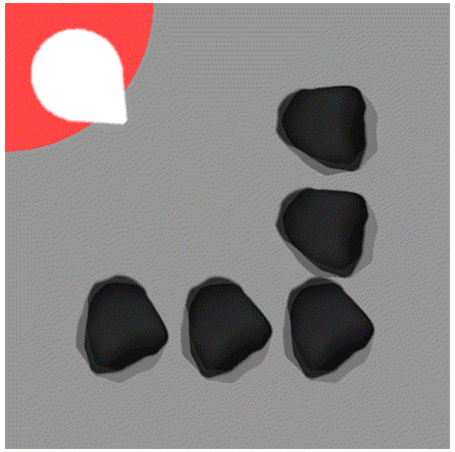
- Continuous Actions
  - MOTD3 – Multi-objective TD3

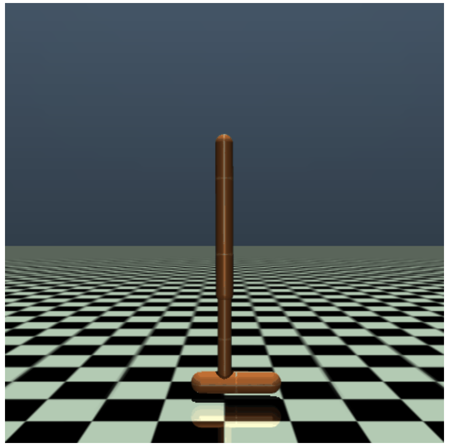
$$\nabla J(\phi; \mathbf{w}) = \nabla_a \mathbf{Q}_\theta(s, a, \mathbf{w}) \cdot \mathbf{w}|_{a=\pi_\phi(s,\mathbf{w})} \nabla_\phi \pi_\phi(s, \mathbf{w})$$

Experiments

- Three environments: Deep Sea Treasure, Minecart, and MO-Hopper
  - Discrete and continuous state and action spaces



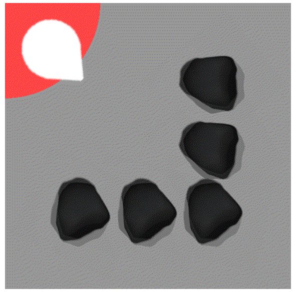




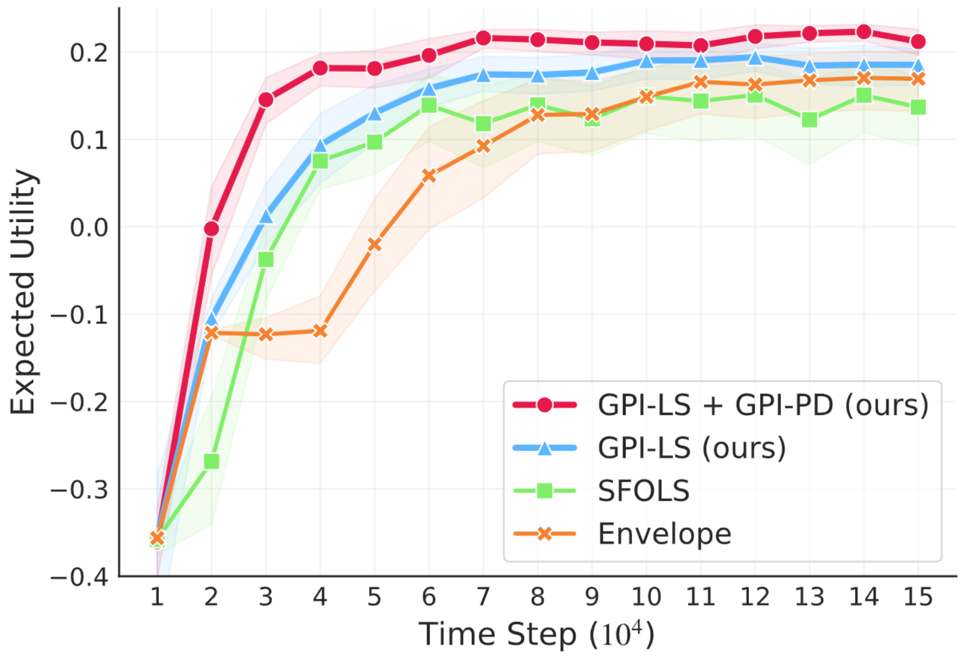
- Evalution metrics: *Expected Utility* (EU) and *Maximum Utility Loss* (MUL)

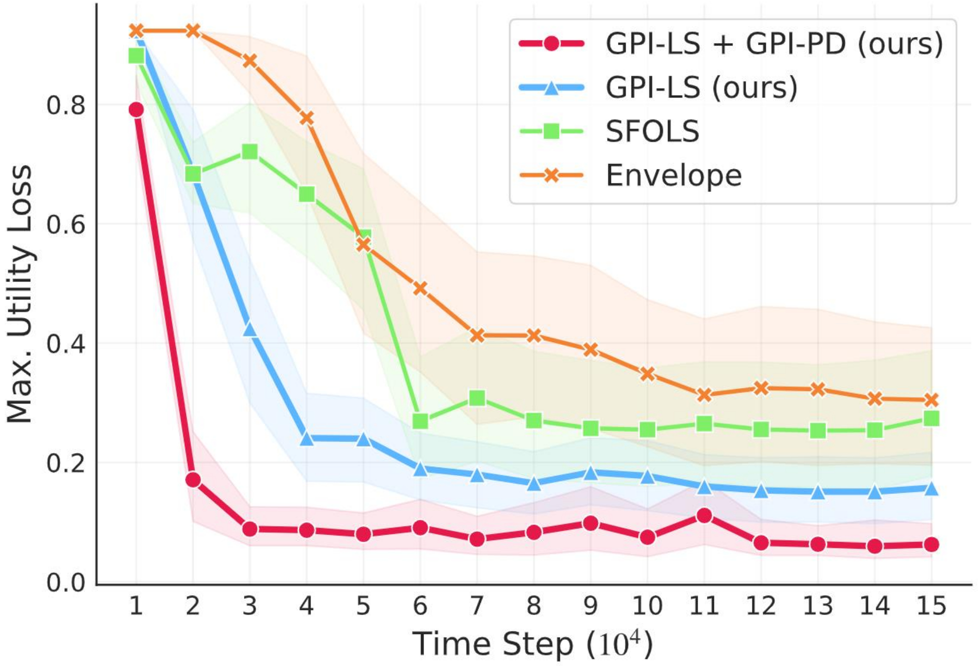
$$\mathrm{EU}(\Pi) = \mathbb{E}_{\mathbf{w}\sim\mathcal{W}}[\max_{\pi\in\Pi} v^{\pi}_{\mathbf{w}}]$$
$$\mathrm{MUL}(\Pi) = \max_{\mathbf{w}\in\mathcal{W}}(v^{*}_{\mathbf{w}} - \max_{\pi\in\Pi} v^{\pi}_{\mathbf{w}})$$

Environments available on MO-Gymnasium: https://github.com/Farama-Foundation/MO-Gymnasium



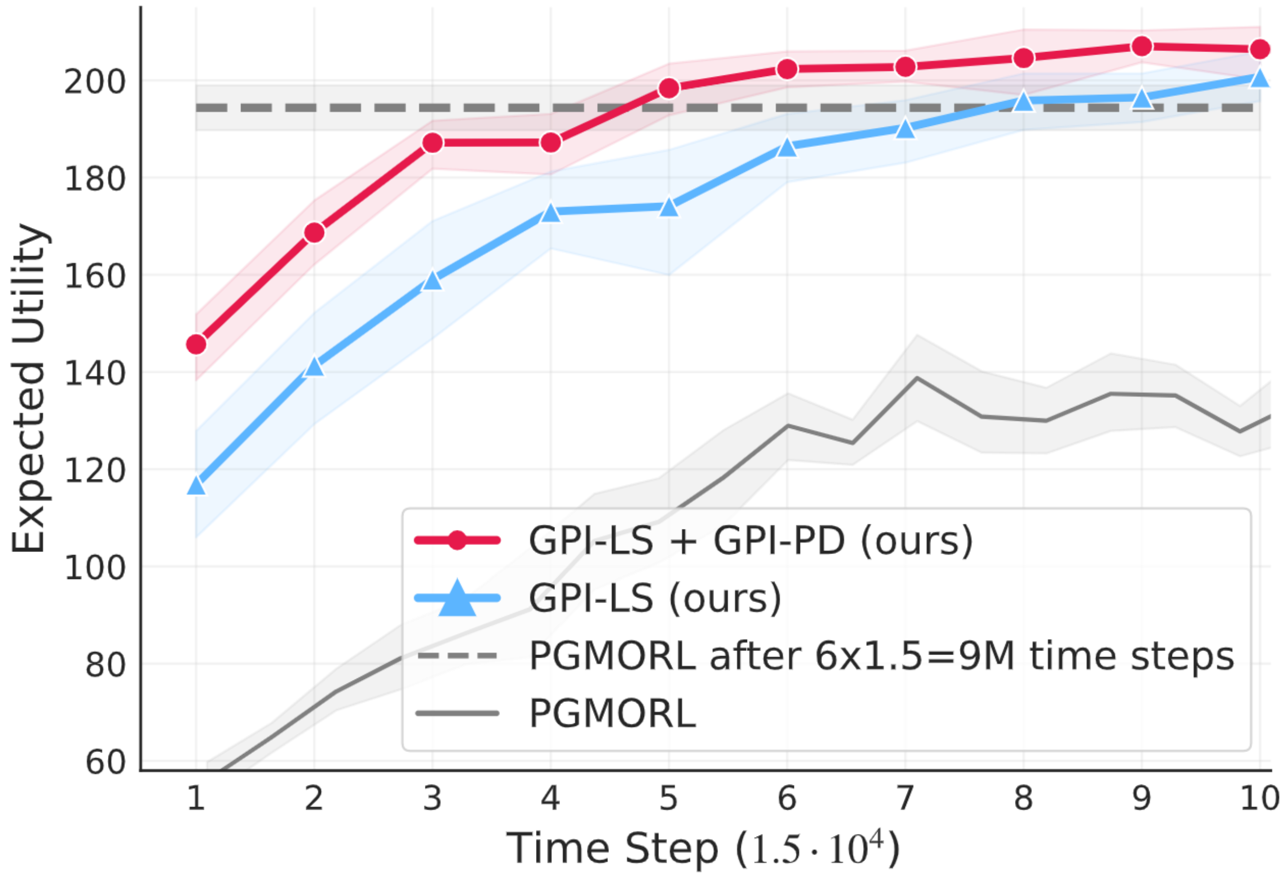
MineCart





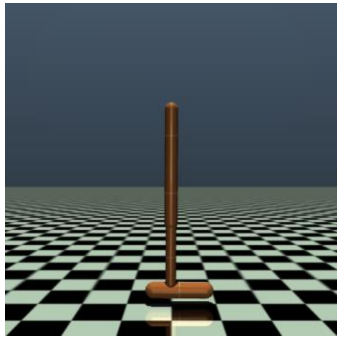
- GPI-LS and GPI-LS+GPI-PD consistently identify near optimal solutions
- Our methods' performance metrics strictly dominate that of competitors

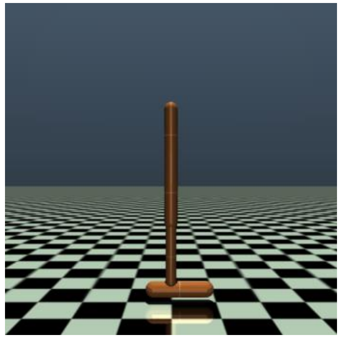
MO Hopper

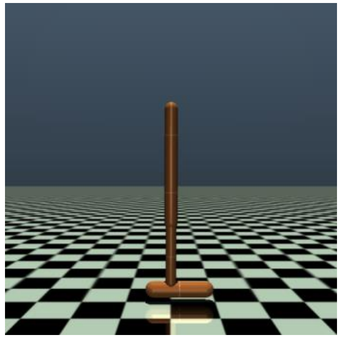


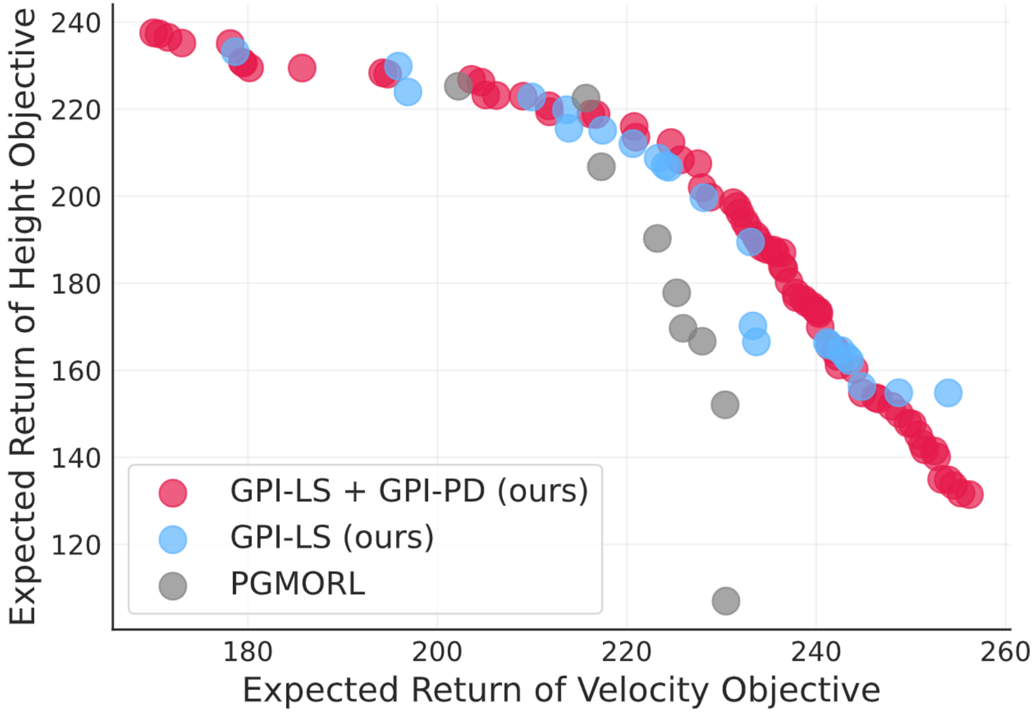
- Our methods achieve higher expected utility and converged to better solutions
- Require *ten times fewer* environment interactions compared to SOTA method

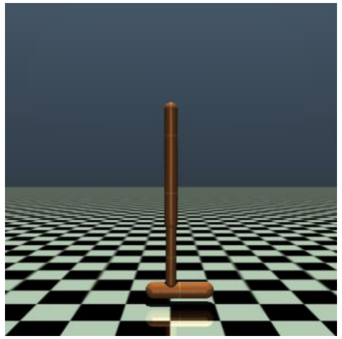
MO Hopper – Pareto Front











Conclusion

- We introduced two principled GPI-based prioritization methods
  - Monotonically improve the quality of the set of policies
  - Guaranteed to identify (near) optimal set of policies
- GPI-PD is the first model-based MORL algorithm for continuous states
- Outperforms state-of-the-art MORL algorithms in challenging tasks
  - Significantly improves sample-efficiency

https://github.com/Farama-Foundation/MO-Gymnasium

Python tests passing | pre-commit enabled | code style black

MO-Gymnasium: Multi-Objective Reinforcement Learning Environments

MO-Gymnasium is an open source Python library for developing and comparing multi-objective reinforcement learning algorithms by providing a standard API to communicate between learning algorithms and environments, as well as a standard set of environments compliant with that API. Essentially, the environments follow the standard Gymnasium API, but return vectorized rewards as numpy arrays.

The documentation website is at mo-gymnasium.farama.org, and we have a public discord server (which we also use to coordinate development work) that you can join here: https://discord.gg/bnJ6kubTg6.

Alegre et al. 2022. MO-Gym: A Library of Multi-Objective Reinforcement Learning Environments. *In Proceedings of the 34th Benelux Conference on Artificial Intelligence BNAIC/Benelearn 2022.*

https://github.com/Farama-Foundation/MO-Gymnasium

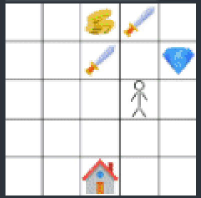
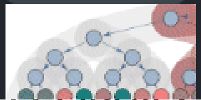
Python tests passing | pre-commit enabled | code style black

MO-Gymna … nt Learning E…

MO-Gymnasium is an … reinforcement learning algorithms by … d environments, as well as a standard set … w the standard Gymnasium API, but re…

The documentation we… (which we also use to coordinate develop…

Environments

Env	Obs/Action spaces	Objectives	Description
`deep-sea-treasure-v0` 	Discrete / Discrete	`[treasure, time_penalty]`	Agent is a submarine that must collect a treasure while taking into account a time penalty. Treasures values taken from Yang et al. 2019.
`resource-gathering-v0` 	Discrete / Discrete	`[enemy, gold, gem]`	Agent must collect gold or gem. Enemies have a 10% chance of killing the agent. From Barret & Narayanan 2008.
`fruit-tree-v0` 	Discrete / Discrete	`[nutri1, ..., nutri6]`	Full binary tree of depth d=5,6 or 7. Every leaf contains a fruit with a value for the nutrients Protein, Carbs, Fats, Vitamins, Minerals and…

Alegre et al. 2022. MO-G… ceedings *of the 34th Benelux Conference on Artificial Intelligence BNAIC/Benelearn 2022.*

https://github.com/LucasAlegre/morl-baselines

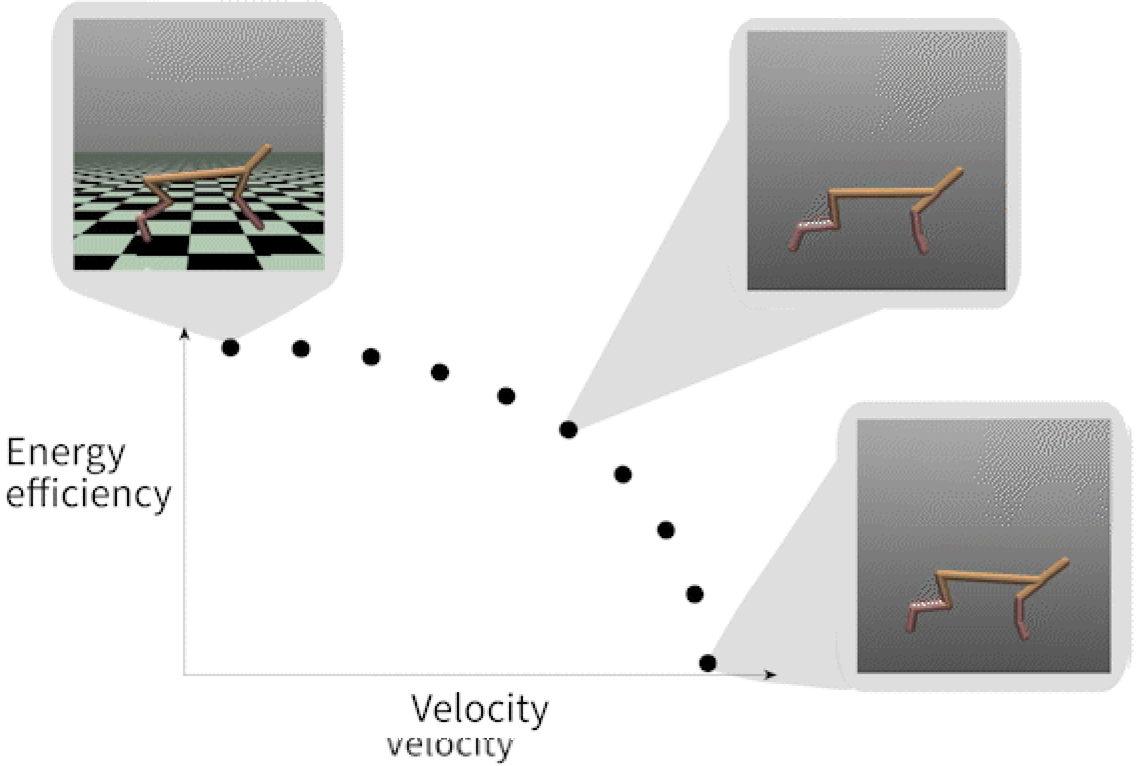
repo status Active | Python tests passing | license MIT | discord 5 online | pre-commit enabled | code style black | imports isort

MORL-Baselines

MORL-Baselines is a library of Multi-Objective Reinforcement Learning (MORL) algorithms. This repository aims at containing reliable MORL algorithms implementations in PyTorch.

It strictly follows MO-Gymnasium API, which differs from the standard Gymnasium API only in that the environment returns a numpy array as the reward.

For details on multi-objective MDP's (MOMDP's) and other MORL definitions, we suggest reading A practical guide to multi-objective reinforcement learning and planning.



Towards Sample-Efficient Multi-Objective Reinforcement Learning

Thank You!

Lucas N. Alegre

lnalegre@inf.ufrgs.br @lnalegre github.com/LucasAlegre/morl-baselines

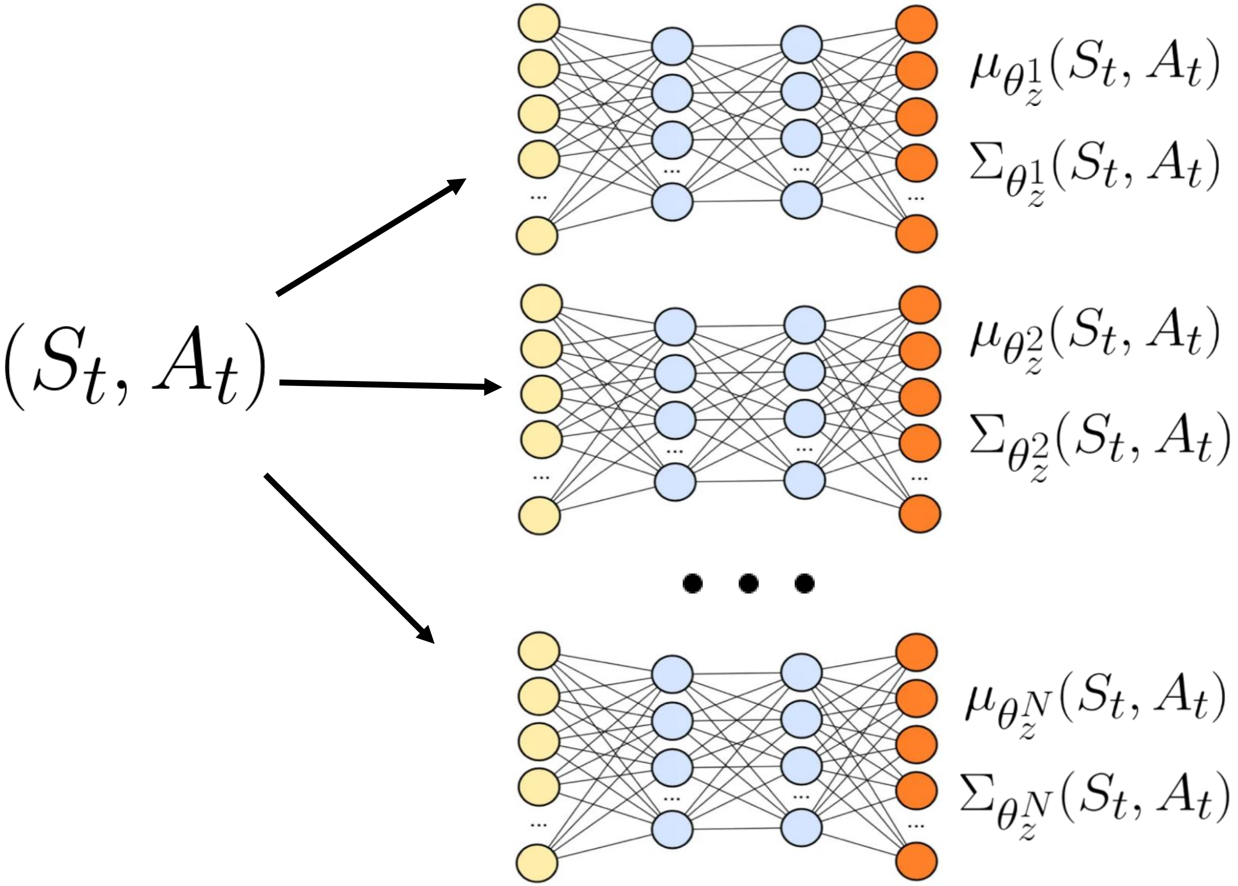
INSTITUTO DE INFORMÁTICA UFRGS

ARTIFICIAL INTELLIGENCE RESEARCH GROUP

Additional Slides

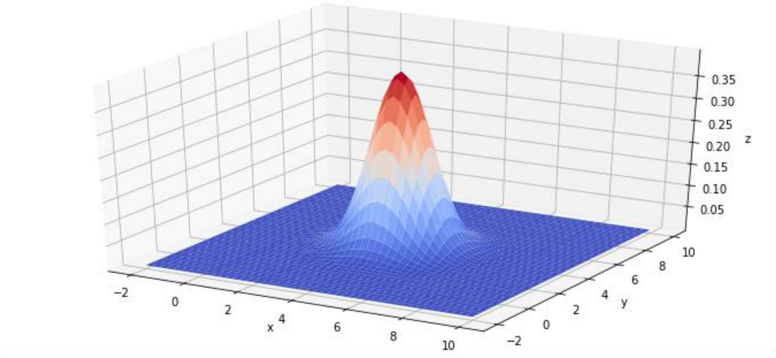
Stochastic Mixture Model of Dynamics

Dynamics approximated via a bootstrap ensemble of probabilistic neural networks [1] parameterized as multivariate Gaussian distribution



$$p_{\theta_z}(S_{t+1}, R_t | S_t, A_t) =$$

$$\mathcal{N}(\mu_{\theta_z}(S_t, A_t), \Sigma_{\theta_z}(S_t, A_t))$$



[1]K.Chua, R.Calandra, R.McAllister, and S. Levine, "*Deep Reinforcement Learning in a Handful of Trials Using Probabilistic Dynamics Models*". (NIPS 2018)

GPI Linear Support

Algorithm 1: GPI Linear Support (GPI-LS)

Input: MOMDP M

$\pi_{\mathbf{w}}, \mathbf{v}^{\pi_{\mathbf{w}}} \leftarrow \text{NewPolicy}(\mathbf{w} = [1, 0, ..., 0]^\top)$

$\Pi \leftarrow \{\pi_{\mathbf{w}}\}, \mathcal{V} \leftarrow \{\mathbf{v}^{\pi_{\mathbf{w}}}\}, \mathcal{M} \leftarrow \{\}$

while *True* **do**

  $\mathcal{W}_{\text{corner}} \leftarrow \text{CornerWeights}(\mathcal{V}) \setminus \mathcal{M}$

  if $\mathcal{W}_{\text{corner}}$ *is empty* **then**

    return $\Pi, \mathcal{V}$; ▷ Found CCS (or ϵ-CCS)

  $\mathbf{w} \leftarrow \arg\max_{\mathbf{w} \in \mathcal{W}_{\text{corner}}} (v_{\mathbf{w}}^{\text{GPI}} - \max_{\pi \in \Pi} v_{\mathbf{w}}^{\pi})$

  $\pi_{\mathbf{w}}, \mathbf{v}^{\pi_{\mathbf{w}}}, \text{done} \leftarrow \text{NewPolicy}(\mathbf{w}, \Pi)$

  if done **then**

    Add $\mathbf{w}$ to $\mathcal{M}$; ▷ Adds $\mathbf{w}$ to support of partial CCS

  Add $\pi_{\mathbf{w}}$ to Π and $\mathbf{v}^{\pi_{\mathbf{w}}}$ to $\mathcal{V}$

  $\Pi, \mathcal{V} \leftarrow \text{RemoveDominated}(\Pi, \mathcal{V})$

Algorithm 2: GPI-Prioritized Dyna (GPI-PD)

```
Initialize action-value function Q_θ(s, a, w), dynamics model p_φ,
  buffers 𝓑 and 𝓑_model, weight support set 𝓜
𝓜 ← extrema weights of 𝓦; w_0 ~ 𝓜
for t = 0...∞ do
    Every N time steps do                         ▷ GPI Linear Support (Alg. 1)
        𝓥 ← evaluate v^{π_w} for all w ∈ 𝓜
        𝓜, 𝓥 ← RemoveDominated(𝓜, 𝓥)
        𝓦_corner ← CornerWeights(𝓥)
        Add to 𝓜 the top-k weight vectors in 𝓦_corner w.r.t.
          arg max_{w∈𝓦_corner} (v_w^GPI − max_{π∈Π} v_w^π)
    if S_t is terminal then
        w_t ~ 𝓜
        S_t ~ μ
    A_t ← π^GPI(S_t; w_t)  (Eq. (11))            ▷ Follow GPI policy
    Execute A_t, observe S_{t+1}, and R_t
    Add (S_t, A_t, R_t, S_{t+1}) to 𝓑 with priority P_{w_t}(S_t, A_t)
    Update model p_φ with experience tuples from 𝓑
    for H Dyna steps do                           ▷ GPI-Prioritized Dyna
        Sample S ~ 𝓑 according to P_{w_t}   (Eq. (10))
        A ← π^GPI(S; w_t); (Ŝ', R̂) ~ p_φ(·|S, A)
        Add (S, A, R̂, Ŝ') to 𝓑_model
    ▷ Update multi-objective Q-function
    for G gradient updates do
        Build mini-batch {(S_i, A_i, R_i, S'_i)}_{i=1}^b with βb tuples from
          𝓑_model and (1 − β)b tuples from 𝓑
        Update Q_θ by minimizing 𝓛(θ; w_t) + 𝓛(θ; w') w.r.t. θ via
          mini-batch gradient descent, where w' ~ 𝓜
        Update priorities P_{w_t} of all pairs (S_i, A_i) in the mini-batch
```