Entropy-Regularized Deep Reinforcement Learning with Linear Programming

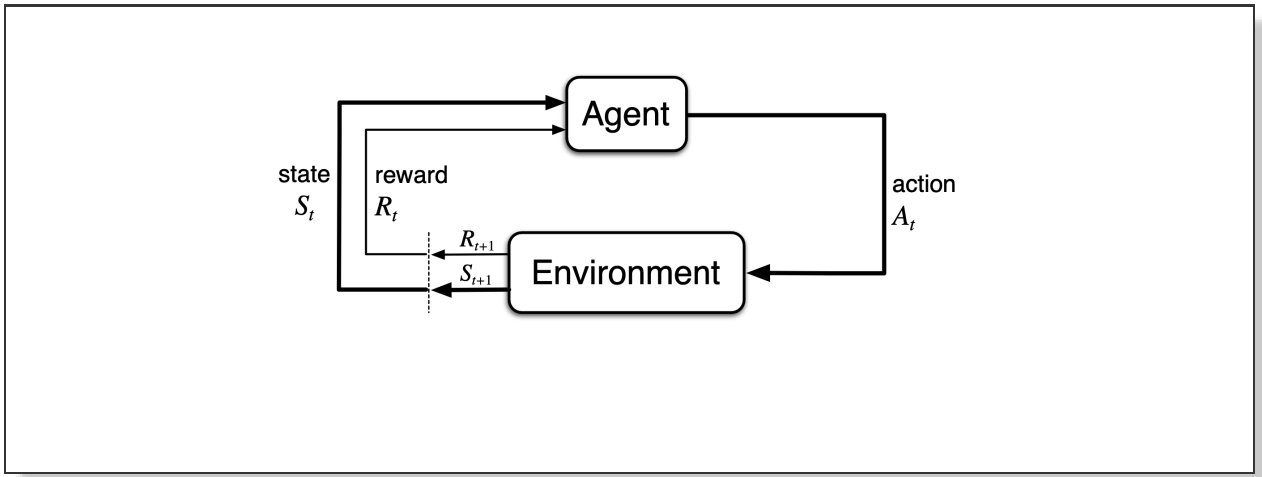
Nicolás Vila Alarcón

School of Engineering
Universitat Pompeu Fabra

July, 2024

Reinforcement Learning

A framework for learning how to interact within an environment based on experience.



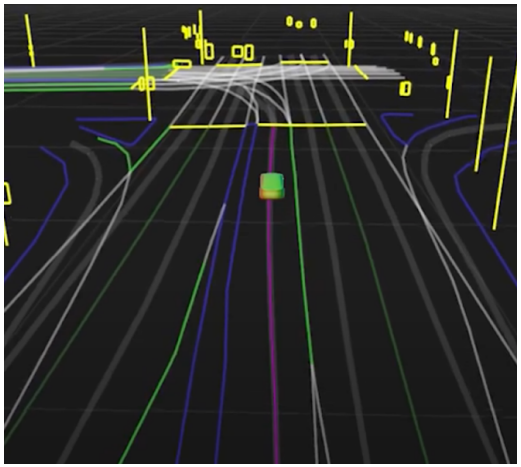
MDP setting [1]

Goal: Maximize the agent's long term reward.

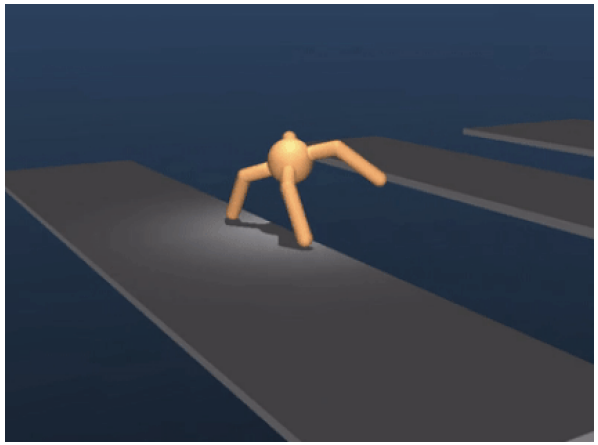
[1] Source: Sutton & Barto

Reinforcement Learning II

Why is this interesting 5



Autonomous car driving[1]

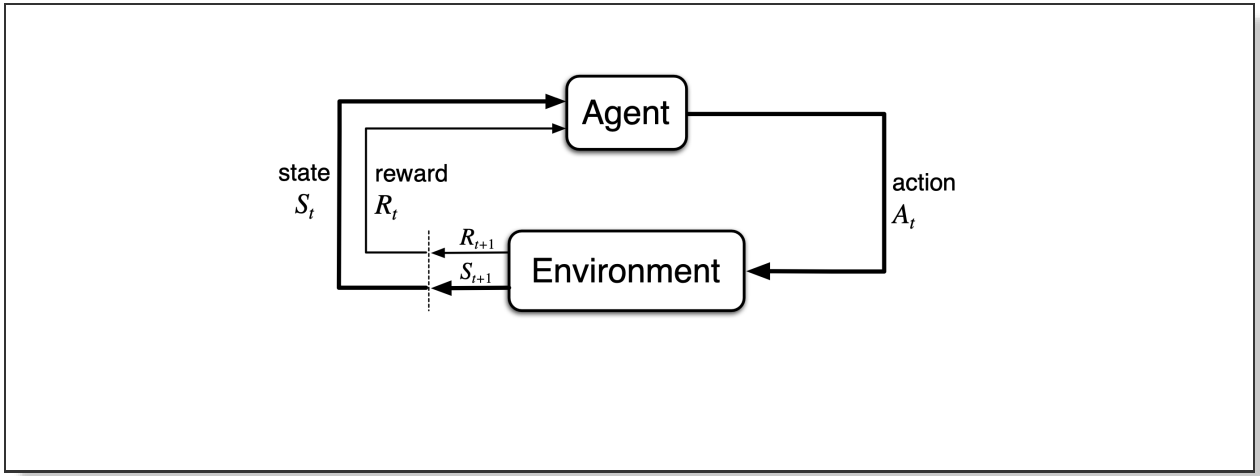


Robot control[2]

DeepMind's AlphaTensor[3]

[1] Source: NVIDIA
[2] Source: DeepMind control library
[3] Source: Nature

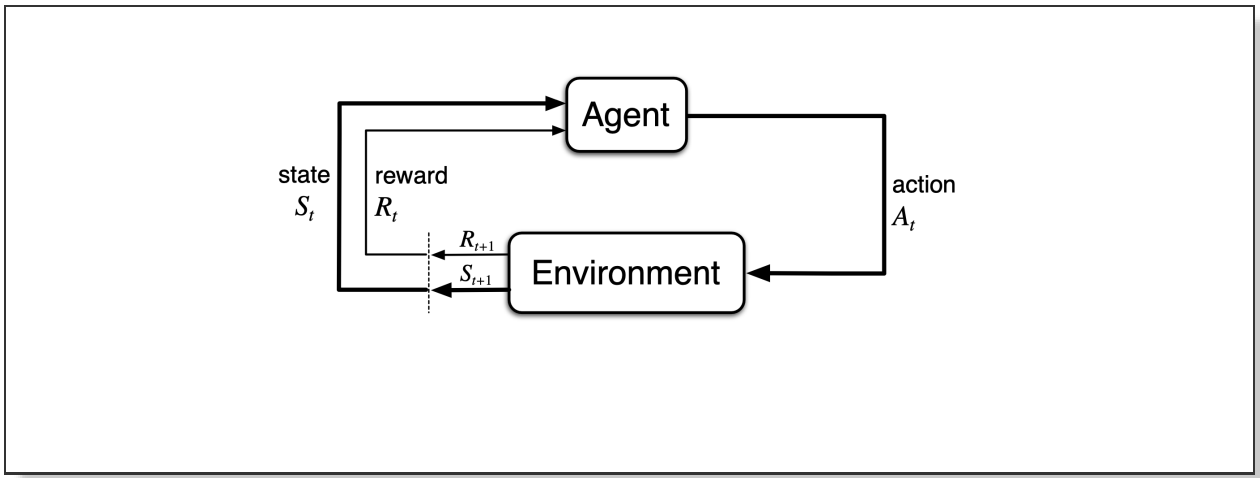
Markov Decision Processes



MDP setting ([1])

- $s_0, a_0, r_1, s_1, \ldots, s_{n-1}, a_{n-1}, r_n, s_n$

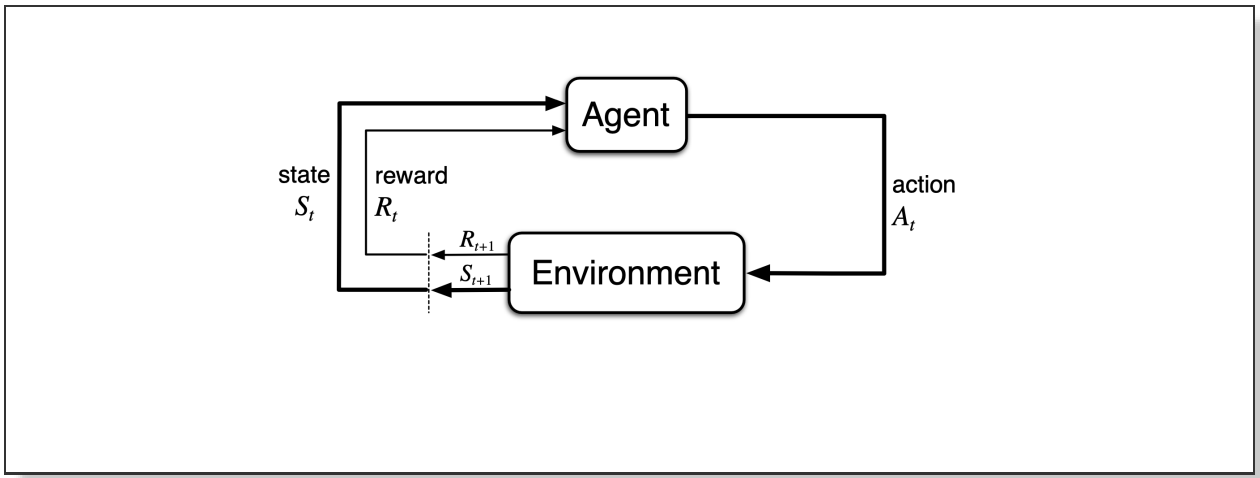
Markov Decision Processes



MDP setting ([1])

- $s_0, a_0, r_1, s_1, \ldots, s_{n-1}, a_{n-1}, r_n, s_n$
- $a_{t+1} \sim \pi(\cdot|s_t)$

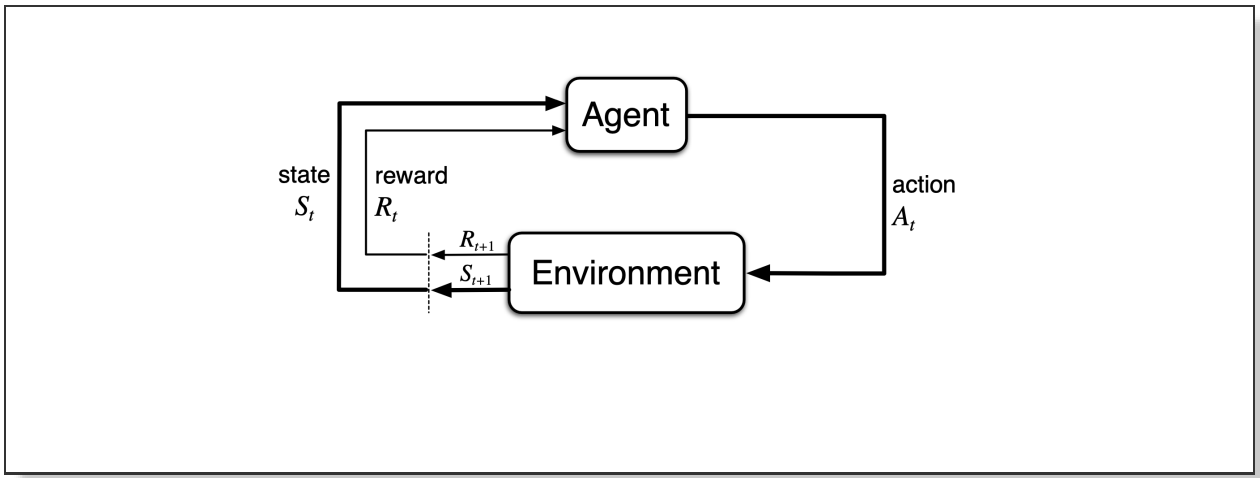
Markov Decision Processes



MDP setting ([1])

- $s_0, a_0, r_1, s_1, \ldots, s_{n-1}, a_{n-1}, r_n, s_n$
- $a_{t+1} \sim \pi(\cdot|s_t)$
- $s_{t+1} \sim P(\cdot|s_t, a_t)$

Markov Decision Processes



MDP setting ([1])

- $s_0, a_0, r_1, s_1, \ldots, s_{n-1}, a_{n-1}, r_n, s_n$
- $a_{t+1} \sim \pi(\cdot|s_t)$
- $s_{t+1} \sim P(\cdot|s_t, a_t)$
- **Goal:** $\max_\pi \mathbb{E}_{a_t \sim \pi, s_0 = s} [\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | \pi, s_0 = s]$

Dynamic Programming

Bellman Optimality Equations ([2])

$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[\max_{a' \in A} Q^*(s',a') \right]$$

- **Value Iteration:** Update the Q-value function based on the reward and future value.

$$\boxed{Q_{k+1}(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[\max_{a'} Q_k(s',a') \right]} \tag{1}$$

$$\pi_{k+1}(a|s) = \begin{cases} 1 & \text{if } a = \arg\max Q_k(s,\cdot) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

- **Alternative notation [15]:**

$$\pi_{k+1} = \arg\max_{\pi} \langle \pi, Q_k \rangle$$

$$Q_{k+1} = r + \gamma P \langle \pi_{k+1}, Q_k \rangle$$

  - $\langle \pi, Q_k \rangle = \sum_{a \in \mathcal{A}} \pi(a|s) Q_k(s,a)$
  - $Pv = \sum_{s'} P(s'|s,a) v(s')$

Regularization in RL

1. Better policy exploration
2. Avoid over-commitment to observed transitions

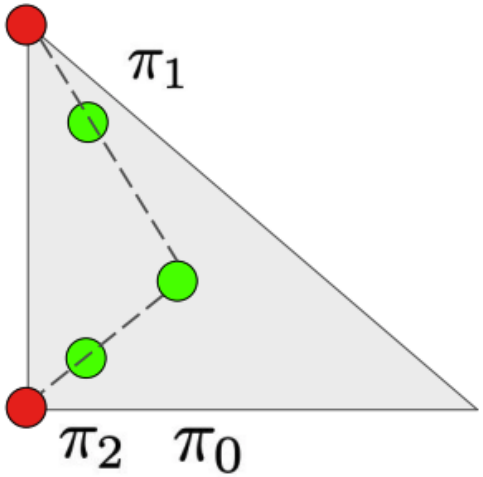


Figure 1: Entropy regularization

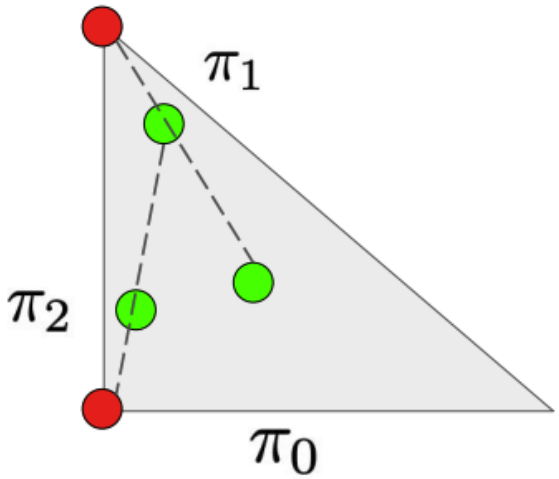


Figure 2: KL regularization

[1] Source: Matthieu Geist lectures

Regularized Dynamic Programming

Legendre-Fenchel Transform

$$\Omega^*(Q) = \max_{\pi} \left(\langle \pi, Q \rangle - \Omega(\pi) \right)$$

$$\nabla\Omega^*(Q) = \arg\max_{\pi} \left(\langle \pi, Q \rangle - \Omega(\pi) \right)$$

Entropy Regularization

- **Policy Update:**

$$\pi_{k+1} = \arg\max_{\pi} \left(\langle \pi, Q_k \rangle + \alpha \mathcal{H}(\pi) \right)$$

$$\pi_{k+1} \propto e^{Q_k/\alpha}$$

$$J(\psi) = \hat{\mathbb{E}}_N \left[\mathbb{E}_{a \sim \pi_\psi(\cdot|s_n)} \left[Q_\theta(s_n, a) - \alpha(\ln \pi_\psi(a|s_n)) \right] \right]$$

- **Q-value Update:**

$$Q_{k+1} = r + \gamma P \left(\langle \pi_{k+1}, Q_k \rangle + \alpha \mathcal{H}(\pi_{k+1}) \right)$$

[1] soft DQN and SAC like updates

[1] [15]

Regularized Dynamic Programming

Legendre-Fenchel Transform

$$\Omega^*(Q) = \max_{\pi} \left(\langle \pi, Q \rangle - \Omega(\pi) \right)$$

$$\nabla\Omega^*(Q) = \arg\max_{\pi} \left(\langle \pi, Q \rangle - \Omega(\pi) \right)$$

KL Regularization

- **Policy Update:**

$$\pi_{k+\mathbf{1}} = \arg\max_{\pi} \left(\langle \pi, Q_k \rangle - \alpha D_{KL}(\pi || \pi_k) \right)$$

$$\pi_{k+\mathbf{1}} \propto \pi_{\mathbf{0}} \exp\left(\sum_{j=\mathbf{0}}^{k} (Q_j / \alpha) \right)$$

$$J(\psi) = \hat{\mathbb{E}}_N \left[\mathbb{E}_{a \sim \pi_\psi(\cdot|s_n)} \left[Q_\theta(s_n, a) - \alpha(\ln \pi_\psi(a|s_n) - \ln \pi_{\psi_{old}}(a|s_n)) \right] \right]$$

- **Q-value Update:**

$$Q_{k+\mathbf{1}} = r + \gamma P \left(\langle \pi_{k+\mathbf{1}}, Q_k \rangle - \alpha D_{KL}(\pi_{k+\mathbf{1}} || \pi_k) \right)$$

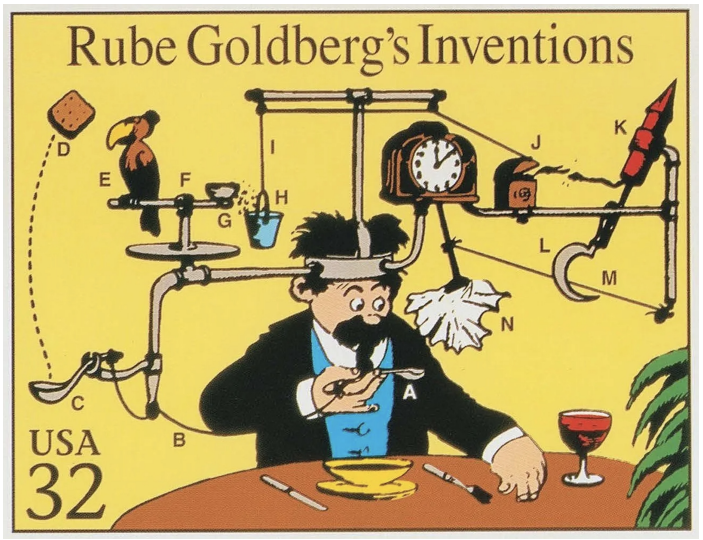
TRPO like policy updates

[1] [15]

Are there other alternatives / Can we do better?

$$\textbf{SBE:} \quad \min_{\theta} \mathbb{E}_{\mathcal{B}} \left[\left(r_i + \gamma \max_{a'} Q_{\bar{\theta}}(s_i', a_i') - Q_{\theta}(s_i, a_i) \right)^2 \right]$$

1. Non-convex & non-smooth objective
2. Unbounded gradients
3. Need heuristic tricks to stabilize training
4. Can run into problems, it might not converge to a good policy



Linear reformulation

$$R_\gamma^\pi = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$$
$$= \sum_{t=0}^{\infty} \mathbb{E}_\pi \left[\gamma^t r(s_t, a_t)\right]$$
$$= \sum_{t=0}^{\infty} \sum_{s,a} \gamma^t P_\pi(s_t = s, a_t = a) r(s, a)$$

[1] images inspired by Gergely Neu's lecture

Linear reformulation

$$R_\gamma^\pi = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$$
$$= \sum_{t=0}^{\infty} \mathbb{E}_\pi \left[\gamma^t r(s_t, a_t)\right]$$
$$= \sum_{t=0}^{\infty} \sum_{s,a} \gamma^t P_\pi(s_t = s, a_t = a) r(s, a)$$
$$= \sum_{s,a} \frac{1}{1-\gamma} (1-\gamma) \sum_{t=0}^{\infty} \gamma^t P_\pi(s_t = s, a_t = a) r(s, a)$$

[1] images inspired by Gergely Neu's lecture

Linear reformulation

$$R_\gamma^\pi = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$$
$$= \sum_{t=0}^{\infty} \mathbb{E}_\pi \left[\gamma^t r(s_t, a_t)\right]$$
$$= \sum_{t=0}^{\infty} \sum_{s,a} \gamma^t P_\pi(s_t = s, a_t = a) r(s, a)$$
$$= \sum_{s,a} \frac{1}{1-\gamma}(1-\gamma) \sum_{t=0}^{\infty} \gamma^t P_\pi(s_t = s, a_t = a) r(s, a)$$
$$= \frac{1}{1-\gamma} \sum_{s,a} \mu_\pi(s, a) r(s, a)$$

- Occupancy measure

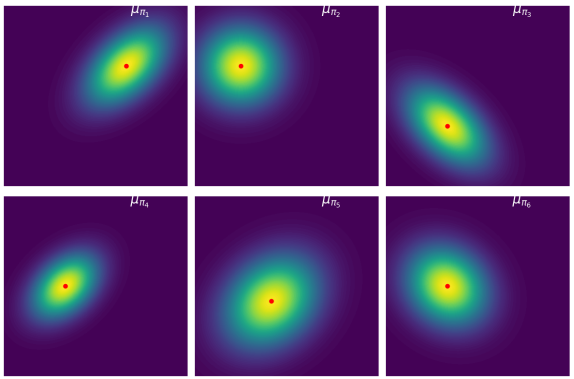
[1] images inspired by Gergely Neu's lecture

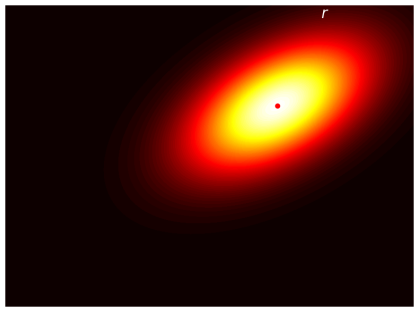
Linear reformulation

$$R_\gamma^\pi = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$$
$$= \sum_{t=0}^{\infty} \mathbb{E}_\pi \left[\gamma^t r(s_t, a_t)\right]$$
$$= \sum_{t=0}^{\infty} \sum_{s,a} \gamma^t P_\pi(s_t = s, a_t = a) r(s, a)$$
$$= \sum_{s,a} \frac{1}{1-\gamma}(1-\gamma) \sum_{t=0}^{\infty} \gamma^t P_\pi(s_t = s, a_t = a) r(s, a)$$

$$= \frac{1}{1-\gamma} \sum_{s,a} \mu_\pi(s, a) r(s, a)$$

- Occupancy measure





[1] images inspired by Gergely Neu's lecture

The Linear Program

Theorem (Manne, 1960)

$$\sum_b \mu(s', b) = \gamma \sum_{s,a} P(s'|s, a)\mu(s, a) + (1 - \gamma)\nu_0$$

The Linear Program

Theorem (Manne, 1960)

$$\sum_b \mu(s', b) = \gamma \sum_{s,a} P(s'|s, a)\mu(s, a) + (1 - \gamma)\nu_0$$

Primal LP

$$\begin{aligned} \underset{\mu \in \mathbb{R}_+^{S \times A}}{\text{maximize}} \quad & \langle \mu, r \rangle \\ \text{subject to} \quad & \sum_b \mu(s', b) = \gamma (P^T \mu)(s') + (1 - \gamma)\nu_0 . \end{aligned}$$

A policy can be extracted from the solution:

$$\pi_\mu(a|s) = \frac{\mu(s, a)}{\sum_{a'} \mu(s, a')}$$

The Linear Program

Theorem (Manne, 1960)

$$\sum_b \mu(s', b) = \gamma \sum_{s,a} P(s'|s, a)\mu(s, a) + (1 - \gamma)\nu_0$$

Primal LP

$$\begin{aligned} \underset{\mu \in \mathbb{R}_+^{S \times A}}{\text{maximize}} \quad & \langle \mu, r \rangle \\ \text{subject to} \quad & \sum_b \mu(s', b) = \gamma (P^T \mu)(s') + (1 - \gamma)\nu_{\mathbf{0}}. \end{aligned}$$

A policy can be extracted from the solution:

$$\pi_\mu(a|s) = \frac{\mu(s, a)}{\sum_{a'} \mu(s, a')}$$

Dual LP

$$\begin{aligned} \underset{V \in \mathbb{R}^S}{\text{minimize}} \quad & (1 - \gamma)\langle \nu_{\mathbf{0}}, V \rangle \\ \text{subject to} \quad & EV \geq r + \gamma PV, \quad \forall s \in S, \forall a \in \mathcal{A} \end{aligned}$$

Practical Algorithms

- Too many constraints & no access to transition function
- Use the LP framework to construct a tractable optimization objective that can work with sample transitions.

Practical Algorithms: Q-REPS

Q-REPS Primal LP ([4])

$$\underset{\mu \in \mathbb{R}_+^{S \times A}}{\text{maximize}} \quad \langle \mu, r \rangle - \eta D_{\mathbf{KL}}(\mu || \mu_{\mathbf{0}}) - \alpha H(d || d_{\mathbf{0}})$$

$$\text{subject to} \quad \sum_b d(s', b) = \gamma \sum_{s,a} P(s'|s,a)\mu(s,a) + (1-\gamma)\nu_{\mathbf{0}},$$

$$\sum_{s,a} d(s,a)\phi(s,a) = \sum_{s,a} \mu(s,a)\phi(s,a)$$

Q-REPS Dual form

$$V_\theta(s) = \alpha \log \left(\sum_a \pi_{\mathbf{0}}(a|s) e^{Q_\theta(s,a)/\alpha} \right)$$

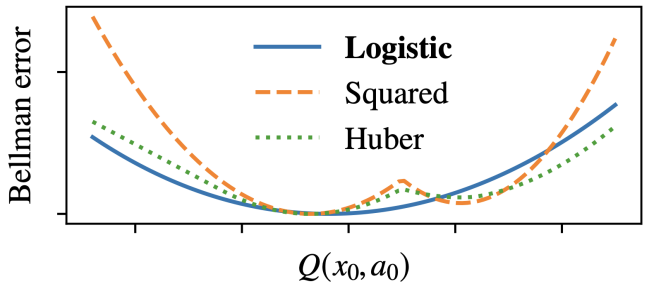
$$\theta^* = \min_\theta \eta \log \mathbb{E}_{s,a \sim \mu_{\mathbf{0}}} \left[e^{(r(s,a) + \gamma \mathbb{E}\left[V_\theta(s')|s,a\right] - Q_\theta(s,a))/\eta} \right] + (1-\gamma)\langle \nu_{\mathbf{0}}, V_\theta \rangle$$

$$\pi^* \propto \pi_{\mathbf{0}}(a|s) e^{\frac{\mathbf{1}}{\alpha}(Q_{\theta*}(s,a) - V_{\theta*}(s))}$$

Practical Algorithms: Q-REPS

$$\hat{\mathcal{G}}(\theta) = \eta \log \frac{1}{N} \sum_n \left[e^{(r(s_n,a_n)+\gamma V_\theta(s'_n)-Q_\theta(s_n,a_n))/\eta} \right] + (1-\gamma)\langle \nu_0, V_\theta \rangle$$

- Convex and smooth objective with bounded gradients
- Tractable policy updates
- Can be estimated with small bias



MinMax-Q-REPS

$$\hat{S}_k(\theta, z) = \sum_n z(n) \left(\hat{\Delta}_\theta(s_n, a_n, s'_n) - \eta \ln(Nz(n)) \right) + (1-\gamma)(\nu_{\mathbf{0}}, V_\theta)$$

$$\min_\theta \hat{\mathcal{G}}(\theta) = \min_\theta \max_z \hat{S}_k(\theta, z)$$

Practical Algorithms: Q-REPS

ELBE Q-REPS

Initialize Q_θ with random parameters θ;
for $k = 0, 1, 2, \ldots, K-1$ **do**
  Run π and collect N sample transitions $\{(s_n, a_n, r_n, s'_n)\}_{n=\mathbf{1}}^{N}$
  for $t = 1, 2, \ldots, T$ **do**
    $\theta_{k,t+\mathbf{1}} \leftarrow \theta_{k,t} - \rho \nabla_\theta \hat{\mathcal{G}}(\theta)$
  end for
  $\theta_k = \frac{\mathbf{1}}{T} \sum_t \theta_{k,t}$
  $\pi_{k+\mathbf{1}}(a|s) \propto \pi_{\mathbf{0}}(a|s) e^{\frac{\mathbf{1}}{\alpha} \sum_{j=\mathbf{0}}^{k} (Q_{\theta_j}(s,a) - V_{\theta_j}(s))}$
end for

MinMax-Q-REPS

Initialize Q_θ with random parameters θ
for $k = 0, 1, 2, \ldots, K-1$ **do**
  Run π and collect N transitions $\{(s_n, a_n, r_n, s'_n)\}_{n=\mathbf{1}}^{N}$
  Initialize $z_{k,\mathbf{0}} \sim \mathcal{U}(N)$
  for $t = 1, 2, \ldots, T$ **do**
    $\theta_{k,t+\mathbf{1}} \leftarrow \theta_{k,t} - \rho \hat{g}_{k,t}$
    $h_{k,t}(n) = \hat{\Delta}_\theta(s_n, a_n, s'_n) - \eta \ln(Nz(n))$
    $z_{k,t+\mathbf{1}}(n) \propto z_{k,t}(n) e^{(\beta h_{k,t}(n))}$
  end for
  $\theta_k = \frac{\mathbf{1}}{T} \sum_t \theta_{k,t}$
  $\pi_{k+\mathbf{1}}(a|s) \propto \pi_{\mathbf{0}}(a|s) e^{\frac{\mathbf{1}}{\alpha} \sum_{j=\mathbf{0}}^{k} (Q_{\theta_j}(s,a) - V_{\theta_j}(s))}$
end for

Practical Algorithms: primal-dual API

Primal LP

$$\begin{aligned} \underset{\mu \in \mathbb{R}_+^{S \times A}}{\text{maximize}} \quad & \langle \mu, r \rangle \\ \text{subject to} \quad & \sum_b \mu(s', b) = \gamma (P^T \mu)(s') + (1-\gamma)\nu_{\mathbf{0}}. \end{aligned}$$

Dual LP

$$\begin{aligned} \underset{V \in \mathbb{R}^S}{\max} \quad & (1-\gamma)\langle \nu_{\mathbf{0}}, V_\pi \rangle \\ \text{subject to} \quad & Q = r_\pi + \gamma P V_\pi, \quad \forall s \in S, \forall a \in \mathcal{A} \\ & V_\pi = \mathcal{M}_\pi Q, \quad \forall s \in \mathcal{S} \end{aligned}$$

$\mathcal{M}_\pi Q = \sum_a \pi(a|s) Q_\pi(s,a)$.

Lagrangian Duality: saddle-point problem

$$\begin{aligned} Q_\theta &= f(\theta) \\ \Delta_\theta(s,a,s') &= r(s,a) + \gamma P \mathcal{M}_\pi Q_\theta(s') - Q_\theta(s,a) \\ \min_\theta \max_{\mu \in R^+} \mathcal{L}(\mu, \theta) &= \mathbb{E}_{\substack{(s,a)\sim\mu \\ s' \sim P(\cdot|s,a)}} \Big[\mu(s,a)\Delta_\theta(s,a,s') + (1-\gamma)\langle \nu_{\mathbf{0}}, \mathcal{M}_\pi Q_\theta \rangle \Big] \end{aligned}$$

Practical Algorithms: primal-dual API

An approximated solution to the saddle-point problem

Primal-dual policy evaluation subroutine: a two player game

$$\theta_{k+1}, \mu_{k+1} = \min_\theta \max_{\mu \in R^+} \frac{1}{N} \sum_{n=1}^{N} \left[\frac{\mu(s_n, a_n)}{\mu_k(s_n, a_n)} \hat{\Delta}_\theta(s, a, s') \right] + (1-\gamma)\langle \nu_0, \mathcal{M}_\pi Q_\theta \rangle$$

Practical Algorithms: primal-dual API

An approximated solution to the saddle-point problem

Primal-dual policy evaluation subroutine: a two player game

$$\theta_{k+1}, \mu_{k+1} = \min_\theta \max_{\mu \in R^+} \frac{1}{N} \sum_{n=1}^{N} \left[\frac{\mu(s_n, a_n)}{\mu_k(s_n, a_n)} \hat{\Delta}_\theta(s, a, s') \right] + (1-\gamma)\langle \nu_0, \mathcal{M}_\pi Q_\theta \rangle$$

$$\theta_{k+1}, z_{k+1} = \min_\theta \max_{z \in R^+} \frac{1}{N} \sum_{n=1}^{N} \left[z_k(n) \hat{\Delta}_\theta(s, a, s') \right] + (1-\gamma)\langle \nu_0, \mathcal{M}_\pi Q_\theta \rangle$$

Practical Algorithms: primal-dual API

An approximated solution to the saddle-point problem

Primal-dual policy evaluation subroutine: a two player game

$$\theta_{k+1}, \mu_{k+1} = \min_\theta \max_{\mu \in R^+} \frac{1}{N} \sum_{n=1}^{N} \left[\frac{\mu(s_n, a_n)}{\mu_k(s_n, a_n)} \hat{\Delta}_\theta(s, a, s') \right] + (1-\gamma)\langle \nu_0, \mathcal{M}_\pi Q_\theta \rangle$$

$$\theta_{k+1}, z_{k+1} = \min_\theta \max_{z \in R^+} \frac{1}{N} \sum_{n=1}^{N} \left[z_k(n) \hat{\Delta}_\theta(s, a, s') \right] + (1-\gamma)\langle \nu_0, \mathcal{M}_\pi Q_\theta \rangle$$

$$\mathcal{H}(z) = \sum_n z(n) \ln z(n) - z(n)$$

$$\hat{J}(\theta, z) = \frac{1}{N} \sum_n z(n) \hat{\Delta}_\theta(s_n, a_n, s'_n) - \eta \mathcal{H}(z) + (1-\gamma)\langle \nu_0, \mathcal{M}_\pi Q_\theta \rangle$$

Practical Algorithms: primal-dual API

PD-Policy Evaluation (with stochastic gradient descent-ascent updates):

$$z_{k+1}, \theta_{k+1} = \min_\theta \max_{z \in R^+} \hat{J}(\theta, z)$$

PD-Policy Iteration ([5], [6])

$$\pi_{k+1} = \pi_k \ \text{softmax}\ (\tau Q_{\theta_{k+1}})$$

Practical Algorithms: primal-dual API

Primal-Dual API

1: Initialize $\pi_{\mathbf{0}}$, Q_θ;
2: **for** $k = 0, 1, 2, \ldots, K-1$ **do**
3: Run π_k and collect sample transitions $\{(s_n, a_n, r_n, s'_n)\}_{n=\mathbf{1}}^{N}$
4: Initialize $z_{k,\mathbf{0}} = \mathbb{1}_N$
5: **for** $t = 1, 2, \ldots, T$ **do**
6: $\theta_{k,t+\mathbf{1}} \leftarrow \theta_{k,t} - \rho \hat{g}_{k,t}$
7: $h_{k,t}(n) = \hat{\Delta}_\theta(s_n, a_n, s'_n) - \eta \ln(z(n))$
8: $z_{k,t+\mathbf{1}}(n) = z_{k,t}(n) e^{(\beta h_{k,t+\mathbf{1}}(n))}$
9: **end for**
10: $\pi_{k+\mathbf{1}} \propto \pi_k$ softmax $(\tau Q_{\theta_{k+\mathbf{1}}}(s, a))$
11: **end for**

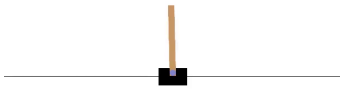
MinMax-Q-REPS

1: Initialize Q_θ with random parameters θ
2: **for** $k = 0, 1, 2, \ldots, K-1$ **do**
3: Run π and collect N transitions $\{(s_n, a_n, r_n, s'_n)\}_{n=\mathbf{1}}^{N}$
4: Initialize $z_{k,\mathbf{0}} \sim \mathcal{U}(N)$
5: **for** $t = 1, 2, \ldots, T$ **do**
6: $\theta_{k,t+\mathbf{1}} \leftarrow \theta_{k,t} - \rho \hat{g}_{k,t}$
7: $h_{k,t}(n) = \hat{\Delta}_\theta(s_n, a_n, s'_n) - \eta \ln(Nz(n))$
8: $z_{k,t+\mathbf{1}}(n) \propto z_{k,t}(n) e^{(\beta h_{k,t}(n))}$
9: **end for**
10: $\theta_k = \frac{\mathbf{1}}{T} \sum_t \theta_{k,t}$
11: $\pi_{k+\mathbf{1}} \propto \pi_k$ softmax $(\tau(Q_{\theta_{k+\mathbf{1}}}(s, a) - V_{\theta_j}(s)))$
12: **end for**

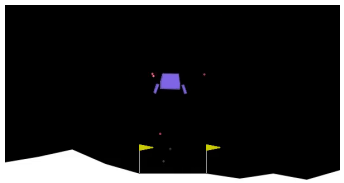
Q-REPS meets NNs

- Parametrize the policy and Q-function with neural networks
- Objective for the policy ?
- Differences between ELBE & Minmax in a practical setting
- Large-scale experiments on 3 different environments

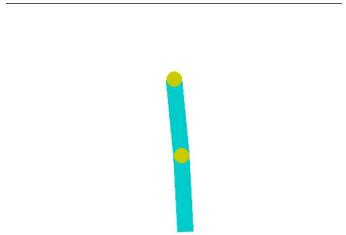
CartPole-v1



LunarLander-v2



Acrobot-v1



[1] https://gymnasium.farama.org/

Policy Parametrization

- **Q-REPS policy update:** $\pi_{k+1}(a|s) \propto \pi_0(a|s) e^{\frac{1}{\alpha} \sum_{j=0}^{k} Q_{\theta,j}(s,a) - V_{\theta,j}(s)}$

KL Loss

$$J(\psi) = \hat{\mathbb{E}}_N \left[\mathbb{E}_{a \sim \pi_\psi(\cdot|s_n)} \left[Q_\theta(s_n, a) - V_\theta(s_n) - \alpha(\ln \pi_\psi(a|s_n) - \ln \pi_{\psi_{\text{old}}}(a|s_n)) \right] \right]$$

Log-Likelihood Loss

$$J(\psi) = \hat{\mathbb{E}}_N \left[\mathbb{E}_{a \sim \pi_\psi(\cdot|s_n)} \left[\ln \pi_\psi(a_n|s_n) e^{(Q_\theta(s_n,a_n) - V_\theta(s_n))/\alpha} \right] \right]$$

Advantadge Estimation

Advantadge: $A_\theta(s,a) = Q_\theta(s,a) - V_\theta(s)$

Temporal Difference

$$G_t = r_t + \gamma V(s_{t+1}) \approx Q^\pi(s_t, a_t)$$
$$\hat{A}_t = G_t - V(s_t)$$

Generalized Advantadge Estimation

$$\hat{A}_t^{\text{GAE}} = (1-\lambda) \sum_{N \geq 0} \lambda^{N-1} \hat{A}_t^N \approx Q^\pi(s_t, a_t)$$

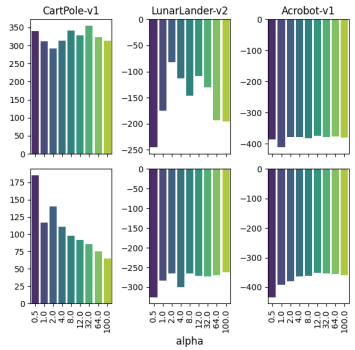
Experimental Design

1. Group together several design choices in the experiment (e.g., regularization coefficients, network architecture, policy losses, etc.)

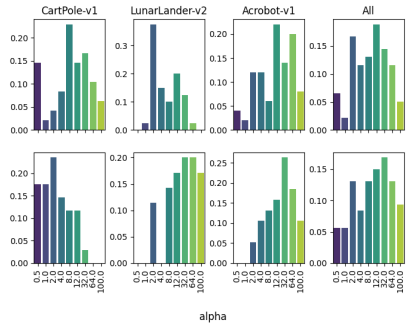
Table 1: Design choices

design choice	default value
`update_epochs`	10
`iteration_size`	10
`minibatch_size`	10
`policy_loss`	KL
`advantadge_estimation`	N-step
α	2
η	2
`learning_rate`	2.5e-4
...	...

95th percentile of reward distribution

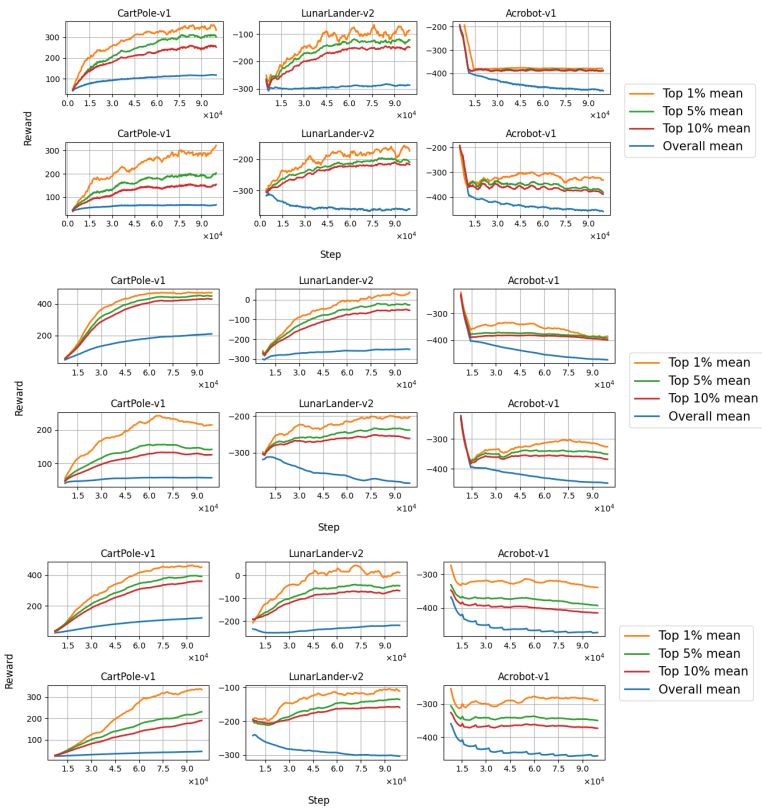


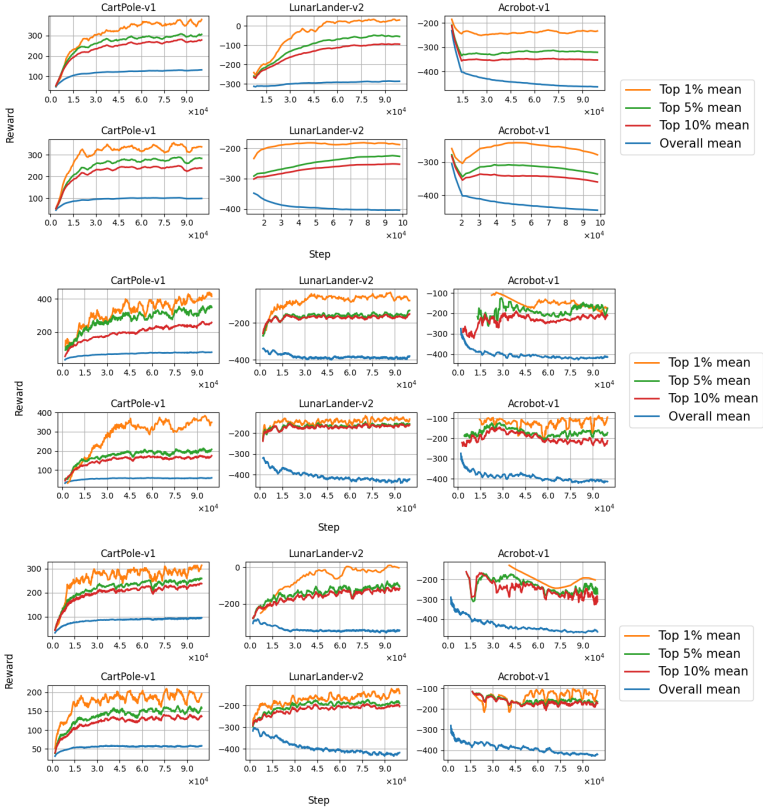
Distribution of choices within the best configurations



Q-REPS experiments

- 30,000 agents trained





Insights

1. Not crucial
2. Important, but environment-dependent
3. Policy loss & GAE

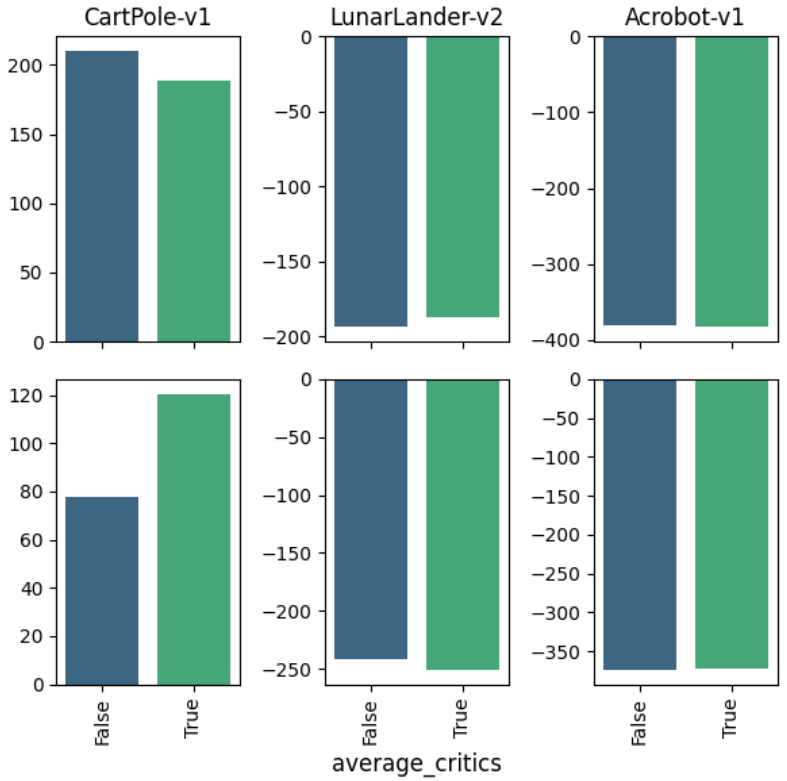


Figure 3: Averaging weights of Q-network

Insights

1. Not crucial
2. Important, but environment-dependent
3. Policy loss & GAE

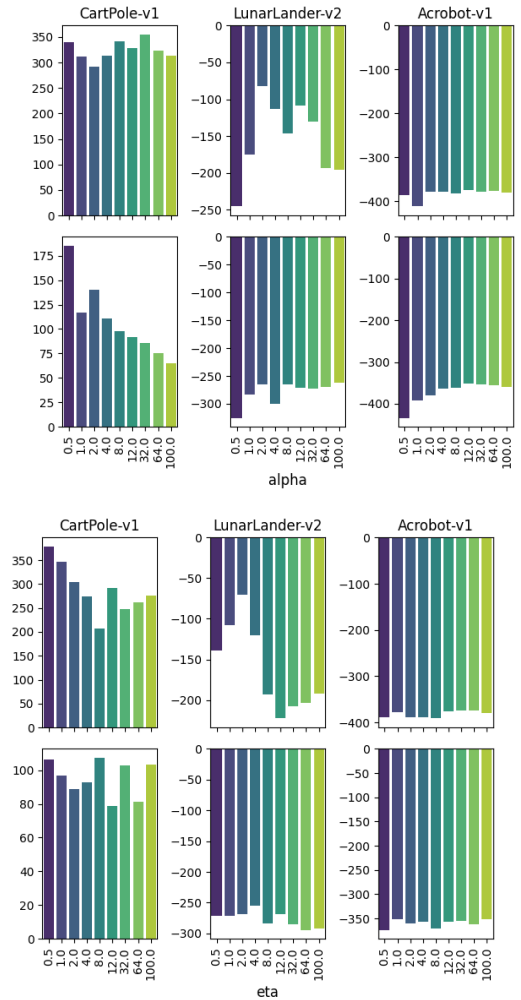


Figure 3: Top: alpha. Bottom: eta

Insights

1. Not crucial
2. Important, but environment-dependent
3. Policy loss & GAE

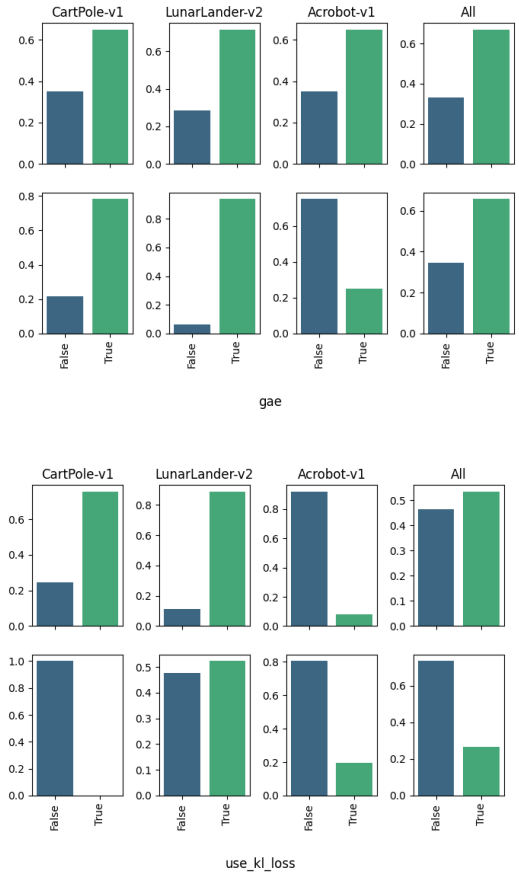


Figure 3: Top: Advantadge estimator. Bottom: Policy loss

Q-REPS Training Curves

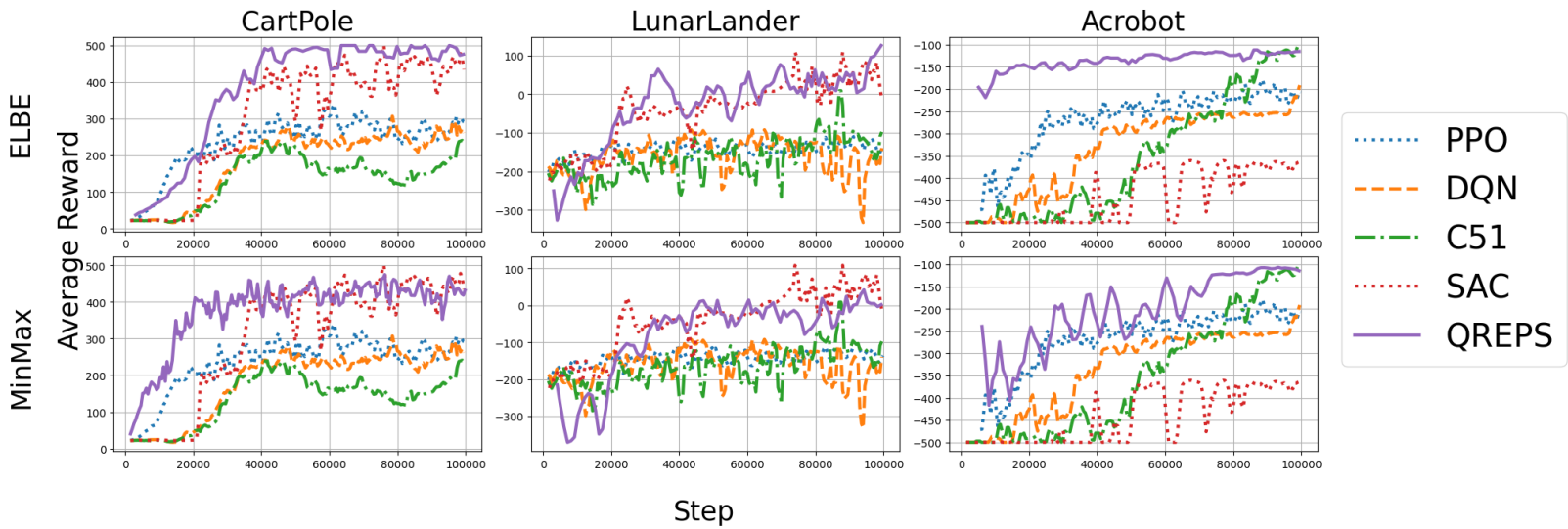
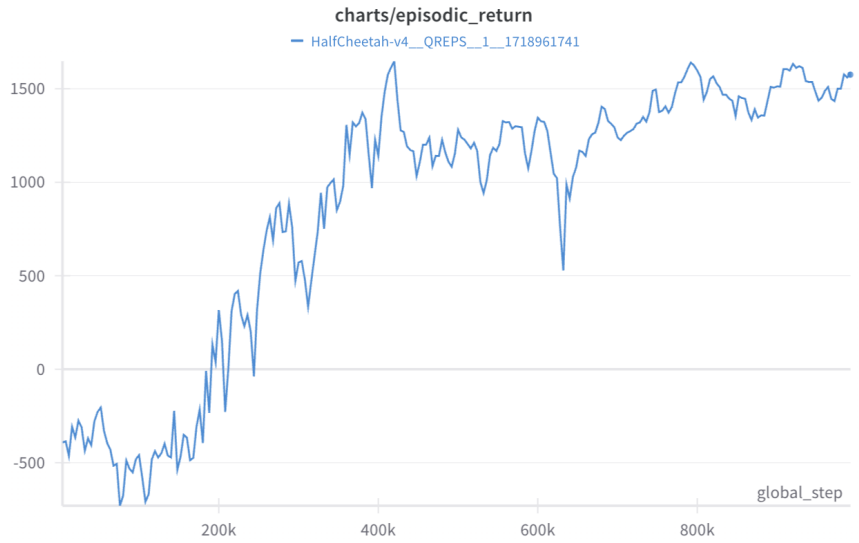


Figure 4: Average episodic return curves

Q-REPS Results

- Could it work with continuous actions?



Tabular PD-API

- Canonical PD : $\mathcal{L}(V, \mu) = \langle \mu, r + \gamma V(s') - EV \rangle + (1 - \gamma)\langle \nu_0, V \rangle$.
- PD-API: $\mathcal{L}(Q, z) = \langle z, r + \gamma \mathcal{M}_\pi Q(s') - Q \rangle + (1 - \gamma)\langle \nu_0, \mathcal{M}_\pi Q(s) \rangle$.
- MinMax Q-REPS: $\mathcal{L}(Q, z) = \langle z, r + \gamma V(s') - Q \rangle + (1 - \gamma)\langle \nu_0, V \rangle$.

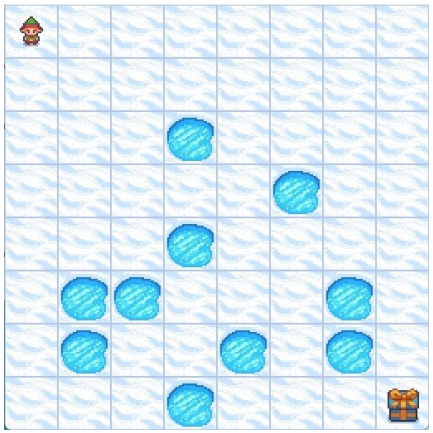


Figure 5: FrozenLake-v1 environemnt

Tabular PD-API II

1. Represent Q as a $\mathcal{S} \times \mathcal{A}$ matrix
2. Build a gradient from transitions (s_n, a_n, s'_n)
3. Update z and Q for a few epochs
4. Update the policy

Convergence curves and tabular policies

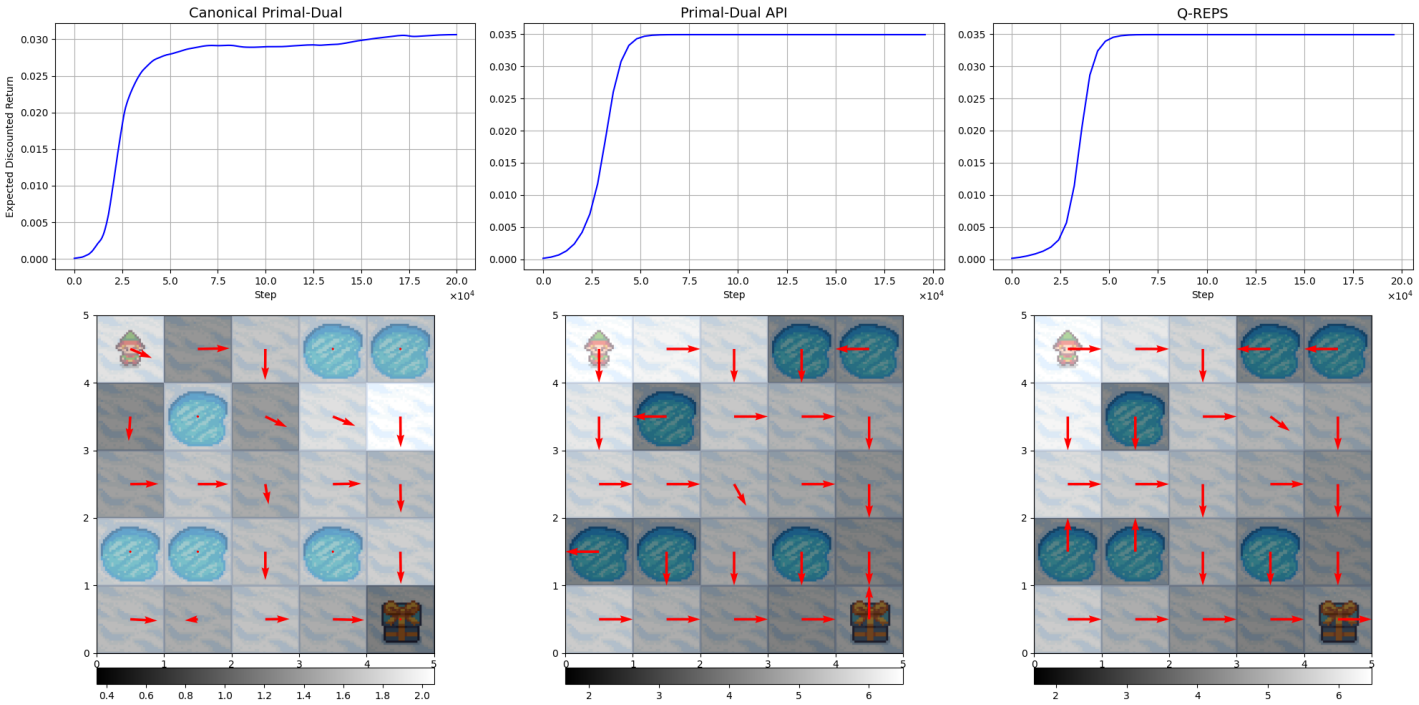


Figure 6: 5x5 grid

Convergence curves and tabular policies

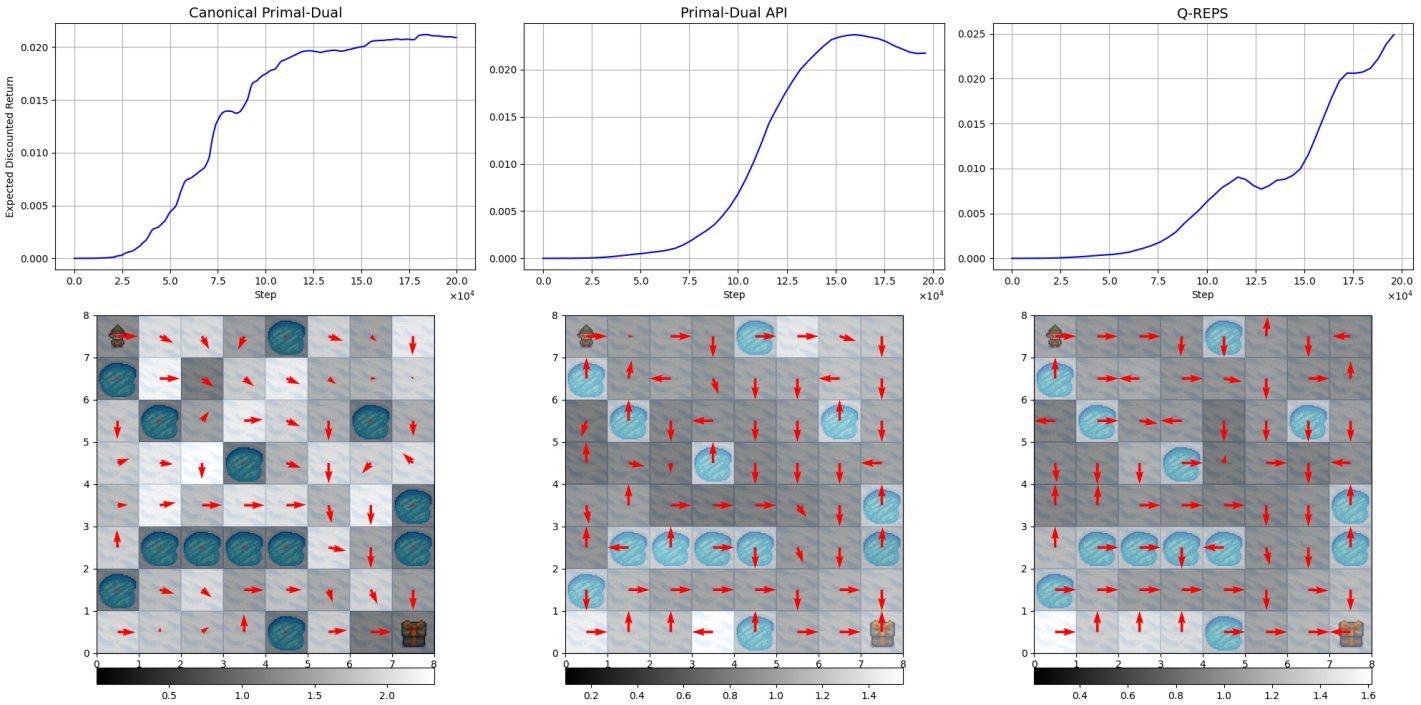


Figure 6: 8x8 grid

Insights

1. Q-REPS & PD-API very similar
2. PD-API more stable
3. Q-REPS usually converges faster
4. Trained 2,000 agents with different configurations
5. Smaller regularization coefficients & larger policy updates for Q-REPS

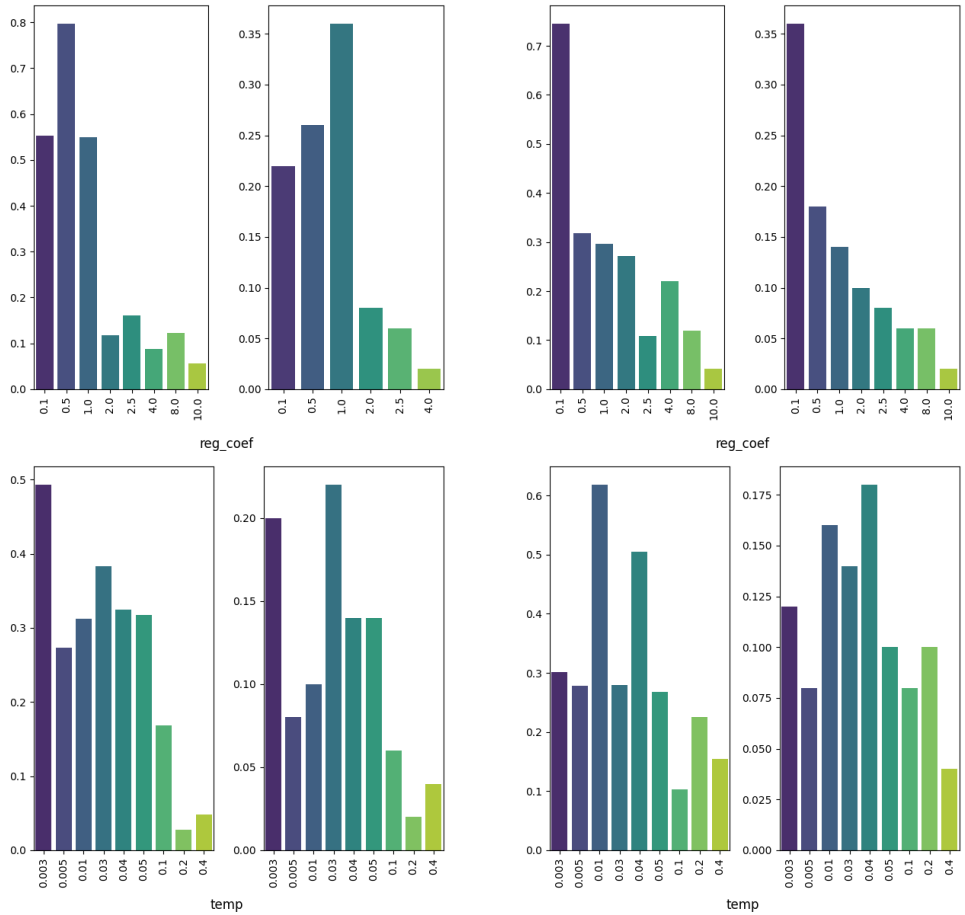


Figure 7: Left: PD-API, Right: Q-REPS

Conclusion

1. Comparable performance to other important Deep RL algorithms
2. Large scale experiments to aid in future algorithms rooted in the LP formulation
3. How would it do in Atari environemnts & continuous environments ?
4. New Primal Dual Policy Evaluation & Iteration algorithm with similar performance to MinMax Q-REPS in a tabular environment.
5. LP has nice theoretical properties & is a promising avenue for developing Deep RL algorithms too

References I

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, second ed., 2018.

[2] R. Bellman, "The theory of dynamic programming," *Bulletin of the American Mathematical Society*, vol. 60, no. 6, pp. 503 – 515, 1954.

[3] A. S. Manne, "Linear programming and sequential decisions," 1960.

[4] J. Bas-Serrano, S. Curi, A. Krause, and G. Neu, "Logistic q-learning," 2021.

[5] Y. Abbasi-Yadkori, P. Bartlett, K. Bhatia, N. Lazic, C. Szepesvari, and G. Weisz, "POLITEX: Regret bounds for policy iteration using expert prediction," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 3692–3702, PMLR, 09–15 Jun 2019.

[6] S. M. Kakade, "A natural policy gradient," in *Advances in Neural Information Processing Systems* (T. Dietterich, S. Becker, and Z. Ghahramani, eds.), vol. 14, MIT Press, 2001.

[7] A. Agarwal, N. Jiang, S. M. Kakade, and W. Sun, *Reinforcement Learning: Theory and Algorithms*. 2021.

[8] G. Neu, A. Jonsson, and V. Gómez, "A unified view of entropy-regularized markov decision processes," 2017.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013.

References II

[10] J. Peters, K. Mulling, and Y. Altun, "Relative entropy policy search," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 24, pp. 1607–1612, Jul. 2010.

[11] J. Liu, S. Liu, and X. Gu, "Soft q network," 2020.

[12] M. Geist, B. Piot, and O. Pietquin, "Is the bellman residual a bad proxy?," 2017.

[13] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," 2017.

[14] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2018.

[15] M. Geist, B. Scherrer, and O. Pietquin, "A theory of regularized markov decision processes," 2019.

[16] M. Andrychowicz, A. Raichuk, P. Stańczyk, M. Orsini, S. Girgin, R. Marinier, L. Hussenot, M. Geist, O. Pietquin, M. Michalski, S. Gelly, and O. Bachem, "What matters in on-policy reinforcement learning? a large-scale empirical study," 2020.

[17] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A research platform for distributed model selection and training," *arXiv preprint arXiv:1807.05118*, 2018.

[18] B. Kegl, "A systematic study comparing hyperparameter optimization engines on tabular data," 2023.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[20] S. Ruder, "An overview of gradient descent optimization algorithms," 2017.

References III

[21] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[23] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry, "Implementation matters in deep rl: A case study on ppo and trpo," in *International Conference on Learning Representations*, 2020.

[24] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2018.

[25] P. Henderson, J. Romoff, and J. Pineau, "Where did my optimum go?: An empirical analysis of gradient descent optimization in policy gradient methods," 2018.

[26] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," 2016.

[27] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," 2017.

[28] S. Huang, R. F. J. Dossa, C. Ye, and J. Braga, "Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms," 2021.

[29] D. Garg, J. Hejna, M. Geist, and S. Ermon, "Extreme q-learning: Maxent rl without entropy," 2023.

References IV

[30] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[31] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 5, pp. 834–846, 1983.

[32] O. C. M. H. Sebastian Markgraf, Philipp Becker, "Validating logistic q-learning," 2022.

[33] R. S. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," in *Advances in Neural Information Processing Systems* (D. Touretzky, M. Mozer, and M. Hasselmo, eds.), vol. 8, MIT Press, 1995.

[34] A. Agarwal, S. M. Kakade, J. D. Lee, and G. Mahajan, "On the theory of policy gradient methods: Optimality, approximation, and distribution shift," 2020.

[35] M. Towers, J. K. Terry, A. Kwiatkowski, J. U. Balis, G. d. Cola, T. Deleu, M. Goulão, A. Kallinteris, A. KG, M. Krimmel, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, A. T. J. Shen, and O. G. Younis, "Gymnasium," Mar. 2023.

Thank You