Maximum A Posteriori Policy Optimisation

Abhinav Dahiya

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., & Riedmiller, M. (2018). M**aximum a posteriori policy optimisation**. *arXiv preprint arXiv:1806.06920*.

Learning Objectives

1. MPO – Overview and Idea
2. MAP – What and why
3. Implementation and Equations
4. Pros and cons
5. Experiments and simulations

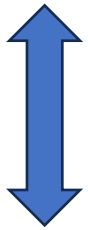
Motivation

- On-policy (TRPO, PPO): robust but poor sample efficiency.
- Off-policy (DDPG, SVG): data-efficient but complex tuning.

Can we get the best of both worlds?

Duality between Control and Estimation

What are the actions which maximize future rewards?



Assuming future success in maximising rewards,
what are the actions most likely to have been taken?

Background

- Markov Decision Processes
  - Define the system as (S,A,T,R,γ)
- Policy representations as neural networks
- Maximum A-Posteriori Estimation
  - How does a prior helps with estimating probability distribution
- KL Divergence

MPO - Overview

- Decompose the objective in terms of the policy parameters θ and an auxiliary variable q.
- Alternating two-phase optimization:
  1. **E-step**:
     - Optimize w.r.t. q
  2. **M-step**:
     - Update θ to maximize expected return

Method Details – Objective Decomposition

$$p(O = 1|\tau) \propto \exp\left(\frac{\sum_t r_t}{\alpha}\right)$$

Probability of the trajectory being optimal

Reward generated via that trajectory

Method Details – Objective Decomposition

$$\log p_\pi(O=1) = \log \int p_\pi(\tau) p(O=1|\tau) d\tau$$

Probability of a policy being optimal

Sum over all trajectories

Method Details – Objective Decomposition

$$\log p_\pi(O=1) = \log \int p_\pi(\tau) p(O=1|\tau) d\tau$$

$$\geq \mathbb{E}_q\Big[\sum_t r_t/\alpha\Big] - \mathrm{KL}\Big(q(\tau)||p_\pi(\tau)\Big) = \mathcal{J}(q,\pi)$$

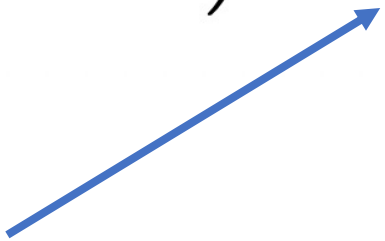
auxiliary (arbitrary) distribution over trajectories

trajectory distribution induced by the policy

Method Details – Objective Decomposition

$$\log p_\pi(O=1) = \log \int p_\pi(\tau) p(O=1|\tau) d\tau$$

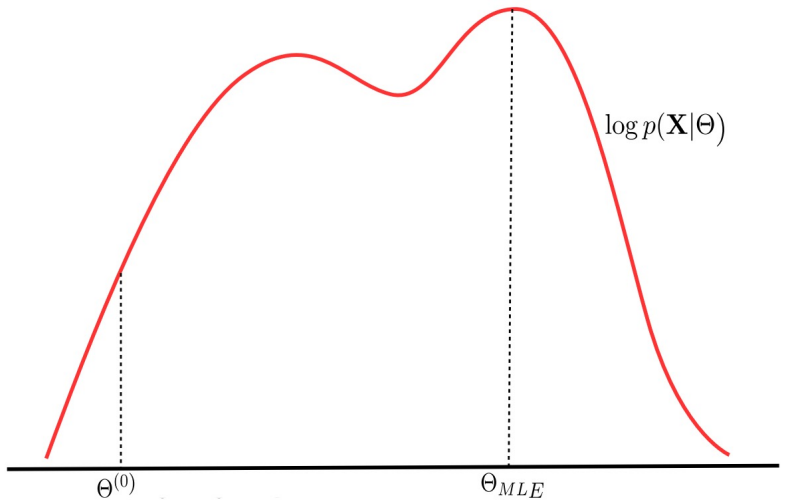
$$\geq \mathbb{E}_q\Big[\sum_t r_t/\alpha\Big] - \mathrm{KL}\Big(q(\tau)||p_\pi(\tau)\Big) = \mathcal{J}(q,\pi)$$



ELBO (Evidence Lower Bound)

What's Going On: A Visual Illustration..

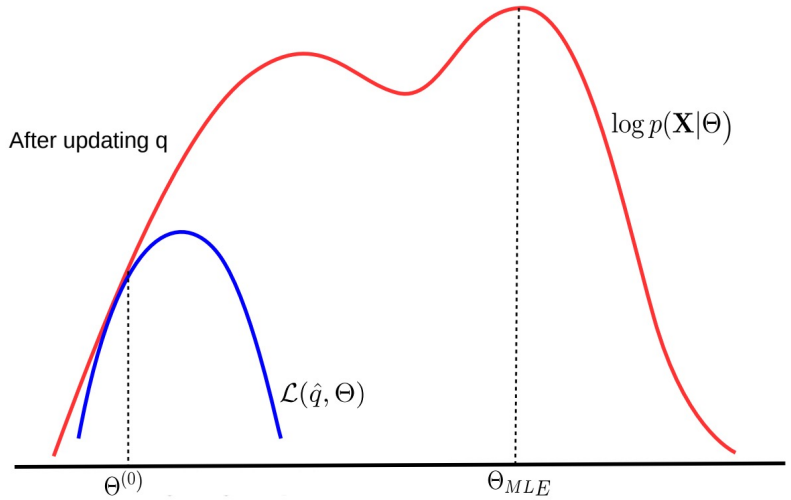
- Step 1: We set $\hat{q} = p(\mathbf{Z}|\mathbf{X}, \Theta^{old})$, $\mathcal{L}(\hat{q}, \Theta)$ touches $\log p(\mathbf{X}|\Theta)$ at Θ^{old}
- Step 2: We maximize $\mathcal{L}(\hat{q}, \Theta)$ w.r.t. Θ (equivalent to maximizing $\mathcal{Q}(\Theta, \Theta^{old})$)



Course Slides – Piyush Rai, IIT Kanpur

What's Going On: A Visual Illustration..

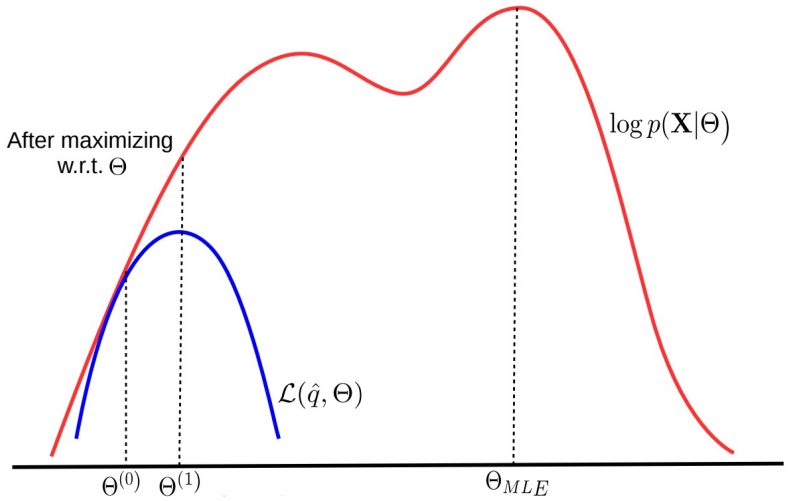
- Step 1: We set $\hat{q} = p(\mathbf{Z}|\mathbf{X}, \Theta^{old})$, $\mathcal{L}(\hat{q}, \Theta)$ touches $\log p(\mathbf{X}|\Theta)$ at Θ^{old}
- Step 2: We maximize $\mathcal{L}(\hat{q}, \Theta)$ w.r.t. Θ (equivalent to maximizing $\mathcal{Q}(\Theta, \Theta^{old})$)



Course Slides – Piyush Rai, IIT Kanpur

What's Going On: A Visual Illustration..

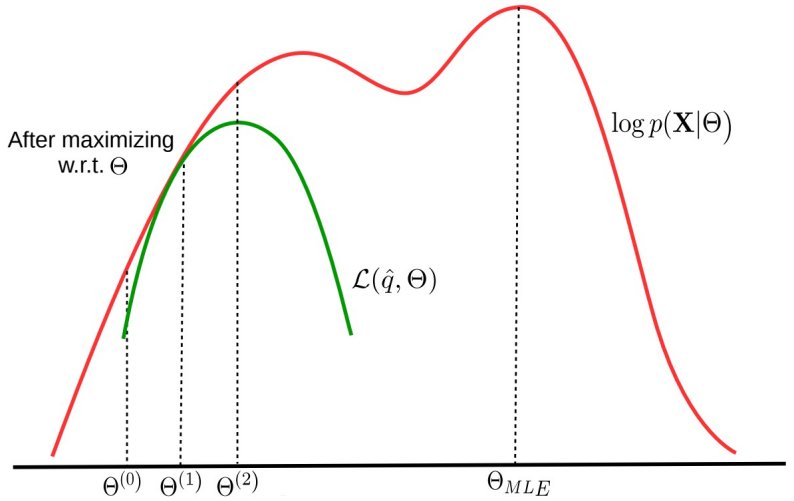
- Step 1: We set $\hat{q} = p(\mathbf{Z}|\mathbf{X}, \Theta^{old})$, $\mathcal{L}(\hat{q}, \Theta)$ touches $\log p(\mathbf{X}|\Theta)$ at Θ^{old}
- Step 2: We maximize $\mathcal{L}(\hat{q}, \Theta)$ w.r.t. Θ (equivalent to maximizing $\mathcal{Q}(\Theta, \Theta^{old})$)



Course Slides – Piyush Rai, IIT Kanpur

What's Going On: A Visual Illustration..

- Step 1: We set $\hat{q} = p(\mathbf{Z}|\mathbf{X}, \Theta^{old})$, $\mathcal{L}(\hat{q}, \Theta)$ touches $\log p(\mathbf{X}|\Theta)$ at Θ^{old}
- Step 2: We maximize $\mathcal{L}(\hat{q}, \Theta)$ w.r.t. Θ (equivalent to maximizing $\mathcal{Q}(\Theta, \Theta^{old})$)



Course Slides – Piyush Rai, IIT Kanpur

Specific Lower Bound

$$\mathcal{J}(q, \boldsymbol{\theta}) = \mathbb{E}_q \Big[\sum_{t=0}^{\infty} \gamma^t \big[r_t - \alpha \mathbf{KL}\big(q(a|s_t) \| \pi(a|s_t, \boldsymbol{\theta})\big) \big] \Big] + \log p(\boldsymbol{\theta})$$

Replace π with θ

Consider specific q

Add a prior

(Partial) E-Step

- Define a regularized Q-function:

$$Q_\theta^q(s,a) = r_0 + \mathbb{E}_{q(\tau), s_0=s, a_0=a}\left[\sum_{t\geq 1}^{\infty} \gamma^t \left[r_t - \alpha \mathrm{KL}(q_t \| \pi_t)\right]\right]$$

- Expand the Q-function via Bellman Operator:

$$\max_q \bar{\mathcal{J}}_s(q, \theta_i) = \max_q T^{\pi,q} Q_{\theta_i}(s,a) = \max_q \mathbb{E}_{\mu(s)}[\mathbb{E}_{q(\cdot|s)}[Q_{\theta_i(s,a)}] - \alpha \mathbf{KL}(q||\pi_i)]$$

- Hard constraint on the KL term:

$$\max_q \mathbb{E}_{\mu(s)}[\mathbb{E}_{q(a|s)}[Q_{\theta_i(s,a)}]] \quad s.t. \mathbb{E}_{\mu(s)}[\mathbf{KL}(q(a|s), \pi(a|s,\theta_i))] < \epsilon$$

- Find *q* that maximizes this Q-function

(Partial) E-Step

$$\max_q \mathbb{E}_{\mu(s)}[\mathbb{E}_{q(a|s)}[Q_{\theta_i}(s,a)]] \quad s.t. \mathbb{E}_{\mu(s)}[\mathrm{KL}(q(a|s), \pi(a|s,\theta_i))] < \epsilon$$

- Solution to this problem can be written as:

$$q_i(a|s) \propto \pi(a|s,\boldsymbol{\theta}_i) \exp\left(\frac{Q_{\theta_i}(s,a)}{\eta^*}\right),$$

where we can obtain η^* by minimising the following convex dual function,

$$g(\eta) = \eta\epsilon + \eta \int \mu(s) \log \int \pi(a|s,\boldsymbol{\theta}_i) \exp\left(\frac{Q_{\theta_i}(s,a)}{\eta}\right) dads$$

E-Step – Summary

- Find the Q-value
- Solve for η^*
- Find value of q_i for the given state-action pair

$$q_i(a|s) \propto \pi(a|s, \boldsymbol{\theta}_i) \exp\left(\frac{Q_{\theta_i}(s,a)}{\eta^*}\right)$$

M-Step

- Update the policy parameter θ

$$\boldsymbol{\theta}_{i+1} = \arg\max_{\boldsymbol{\theta}} \mathcal{J}(q_i, \boldsymbol{\theta})$$

- Recall the expression for ELBO

$$\mathcal{J}(q, \boldsymbol{\theta}) = \mathbb{E}_q \Big[\sum_{t=0}^{\infty} \gamma^t \big[r_t - \alpha \mathbf{KL}\big(q(a|s_t) \| \pi(a|s_t, \boldsymbol{\theta})\big) \big] \Big] + \log p(\boldsymbol{\theta})$$

- Assuming a Gaussian prior around current policy:

$$\max_{\pi} \mathbb{E}_{\mu_q(s)} \Big[\mathbb{E}_{q(a|s)} \Big[\log \pi(a|s, \boldsymbol{\theta}) \Big] - \lambda \mathbf{KL}\Big(\pi(a|s, \boldsymbol{\theta}_i), \pi(a|s, \boldsymbol{\theta}) \Big) \Big]$$

M-Step

$$\max_{\pi} \mathbb{E}_{\mu_q(s)} \Big[\mathbb{E}_{q(a|s)} \Big[\log \pi(a|s, \boldsymbol{\theta}) \Big] - \lambda \mathbf{KL} \Big(\pi(a|s, \boldsymbol{\theta}_i), \pi(a|s, \boldsymbol{\theta}) \Big) \Big]$$

$$\int \mu_q(s) \mathbf{KL}(\pi_i(a|s, \boldsymbol{\theta}), \pi(a|s, \boldsymbol{\theta})) = C_\mu + C_\Sigma,$$

Divergence in mean

$$C_\mu = \int \mu_q(s) \tfrac{1}{2} (\mathrm{tr}(\Sigma^{-1} \Sigma_i) - n + \ln(\frac{\Sigma}{\Sigma_i})) ds,$$

Divergence in covar.

$$C_\Sigma = \int \mu_q(s) \tfrac{1}{2} (\mu - \mu_i)^T \Sigma^{-1} (\mu - \mu_i) ds.$$

M-Step – Summary

1. For the given policy, find C_μ C_Σ
2. Optimize Lagrangian multipliers
3. Optimize Policy parameters

Evaluation

- Continuous control tasks from the DeepMind Control Suite
- Gaussian policy parameterized by neural networks
  - Actions are given by mean and covariance values

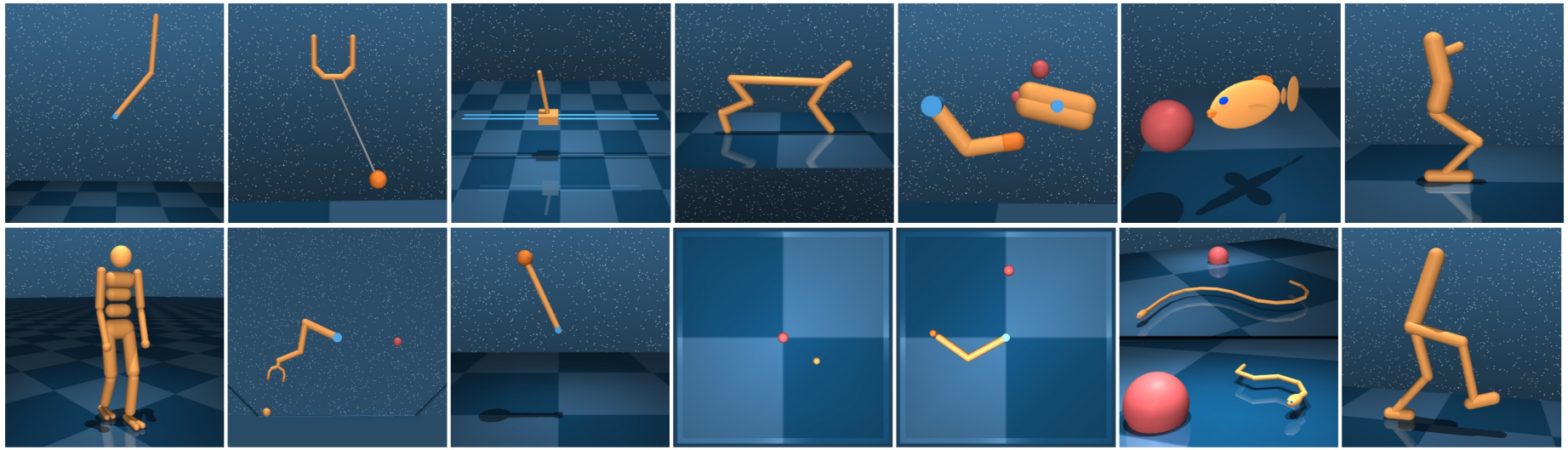
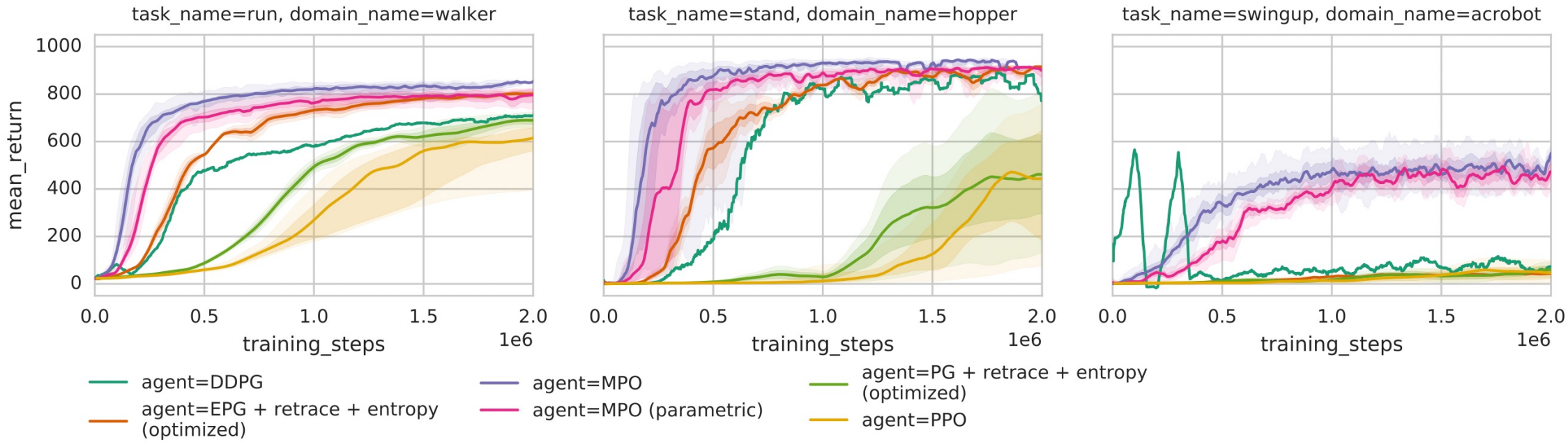


Figure 1: Control Suite domains used for benchmarking. *Top*: Acrobot, Ball-in-cup, Cart-pole, Cheetah, Finger, Fish, Hopper. *Bottom*: Humanoid, Manipulator, Pendulum, Point-mass, Reacher, Swimmers (6 and 15 links), Walker.

Results

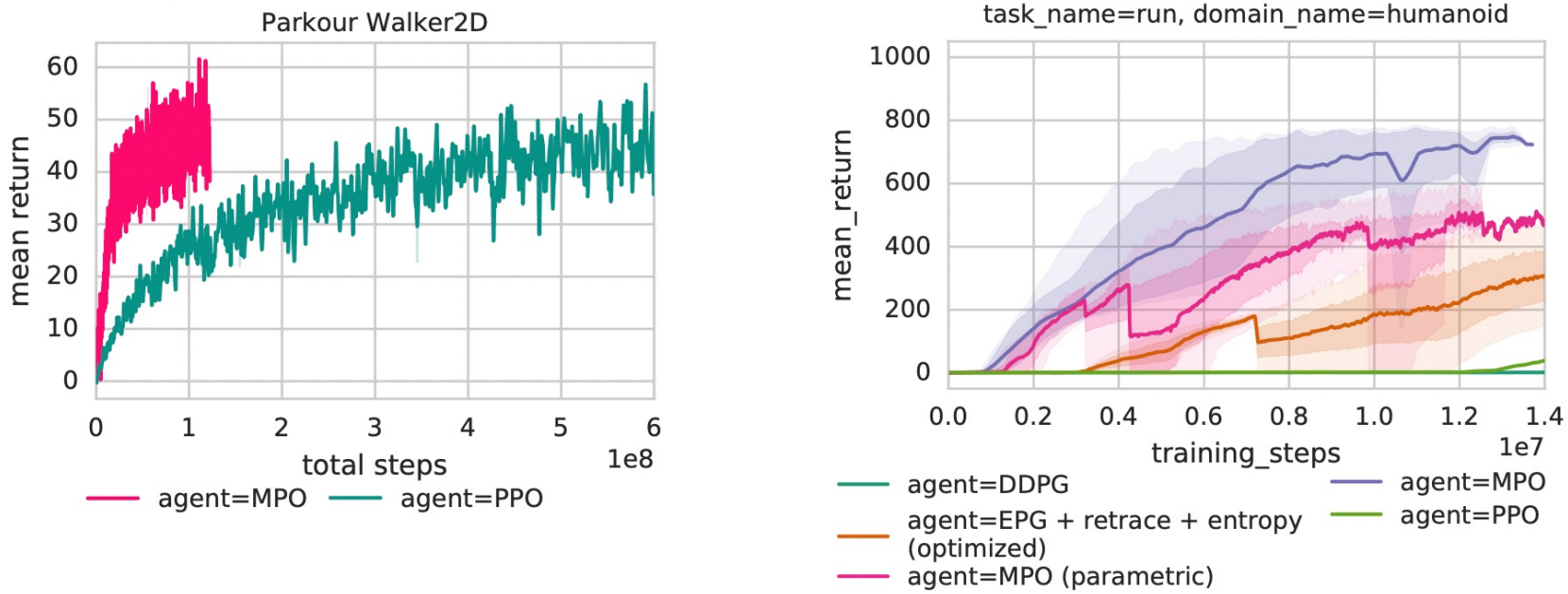
1. Great sample efficiency – Converges faster
2. Stable learning – low variance in policy updates
3. Parametric q distribution reduces efficiency



Results

Similar performance in high-dimensional problems

Figure 3: MPO on high-dimensional control problems (Parkour Walker2D and Humanoid walking from control suite).



Design Choices

1. Adaptive temperature/sensitivity coefficient:
   - Ensures KL constraints are satisfied while maintaining stable learning
2. Flexible and modular design:
   - Free to choose representation of auxiliary distribution
   - Free to choose the Q-learning method
3. Uses exponential reward transformation:
   - Easy to optimize lower bound
4. Decoupled KL constraints:
   - Different learning rates for mean and covariance

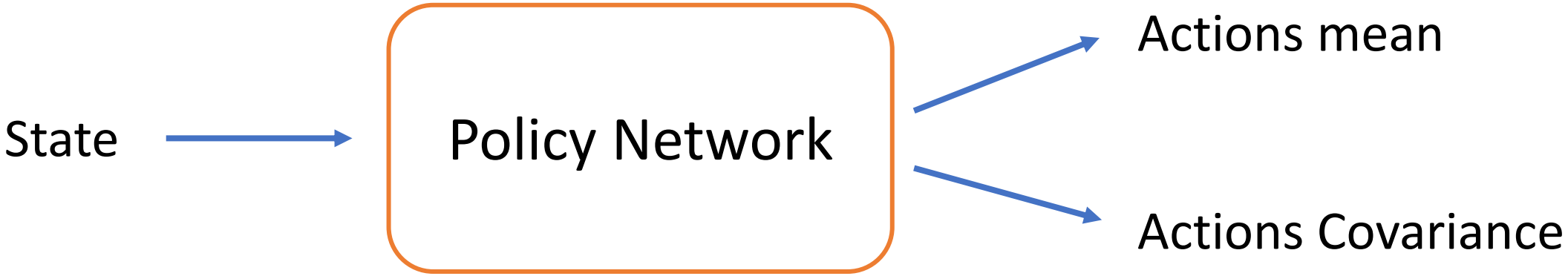
Conclusions

1. MPO – An off-policy reinforcement learning algorithm
2. Connects RL with variational inference
3. Alternating optimization scheme
4. High sample efficiency
5. Robust, stable learning

Part - II

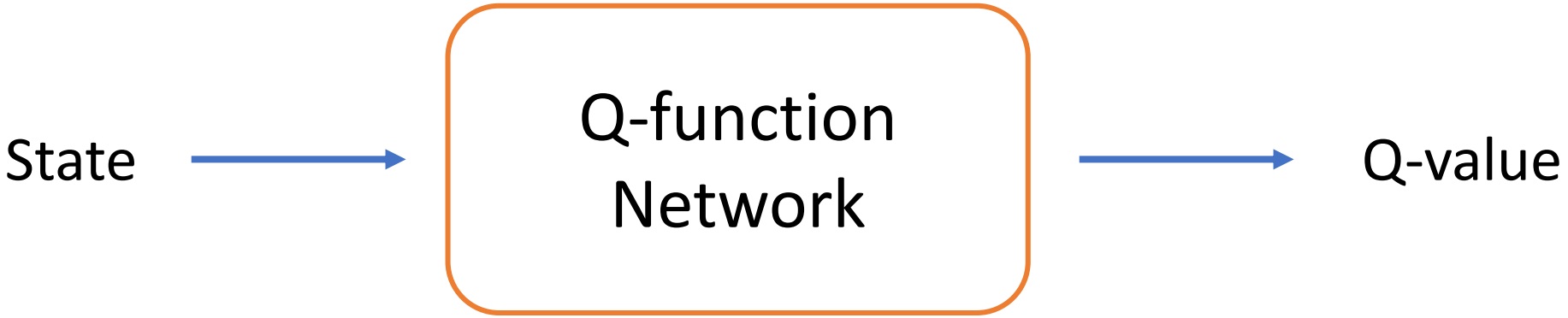
Implementation and Evaluation

Implementation modules



- Policy Network:
  - We assume that the policy is given by a Gaussian distribution
  - Takes in state, outputs action mean and covariance
  - For n actions, output size is $n + \frac{n(n+1)}{2}$
  - 2 Fully connected layers with 50-200 neurons each

Implementation modules



- Q-function Network:
  - Similar structure to the policy network
  - Input state-action pair
  - 1 dimensional output, representing the Q-value
  - 2 Fully connected layers with 50-300 neurons each

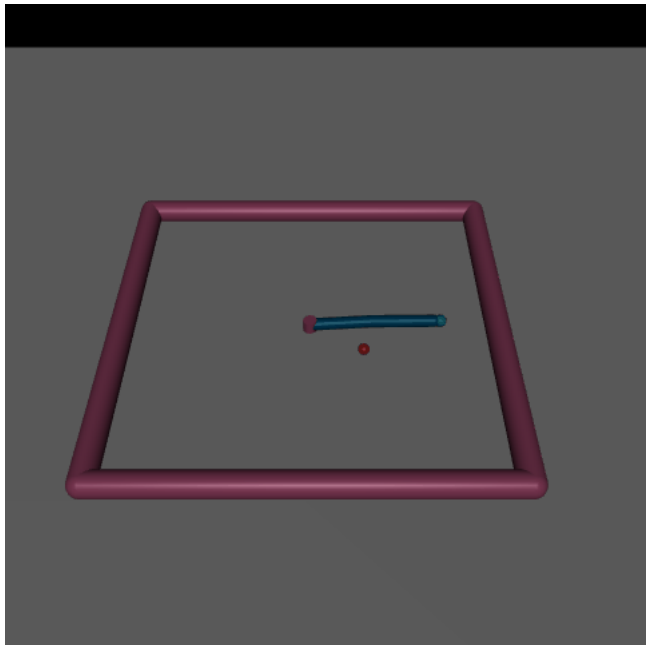
Implementation modifications

- Used Log-sum-exp trick to compute dual coefficient
- Used TD learning to estimate Q-function

Hyperparameters

Parameter	Value
Policy network	100-100 Fully connected layers
Q-function network	200-200 Fully connected layers
Dual constraint ϵ	0.1
Mean constraint ϵ_μ	0.1
Covariance constraint ϵ_Σ	0.0001
Discount factor γ	0.99
Adam learning rate	0.0005

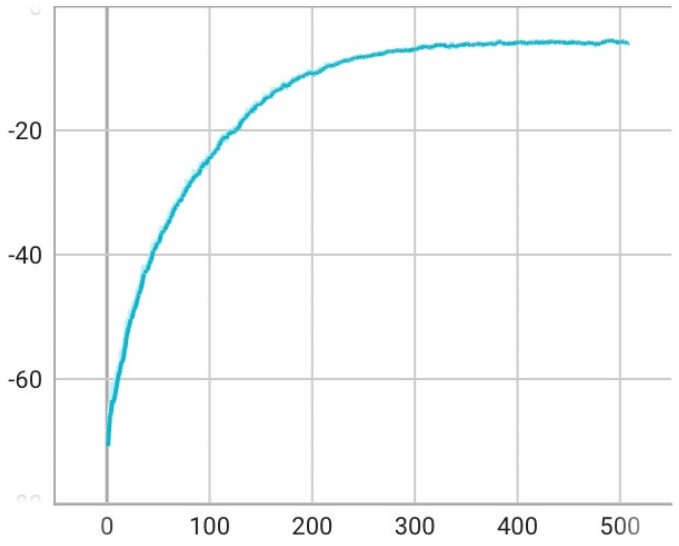
Reacher



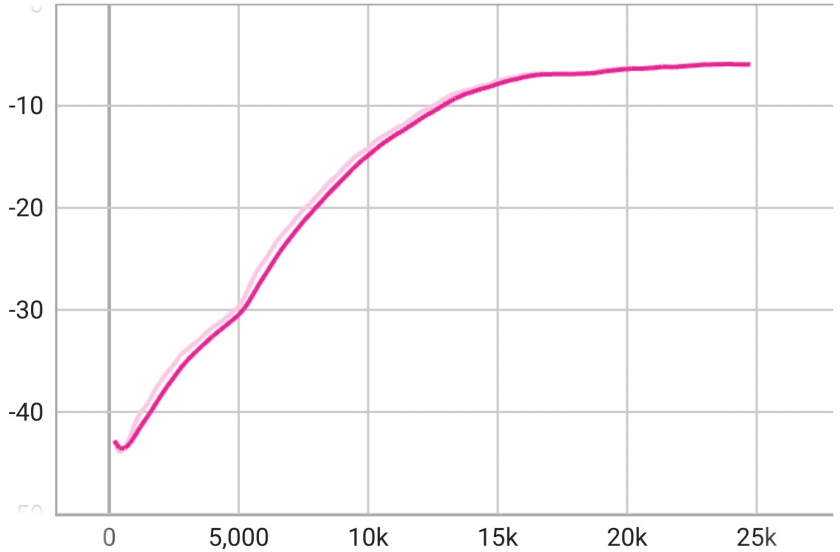
Action Space	`Box(-1.0, 1.0, (2,), float32)`
Observation Space	`Box(-inf, inf, (11,), float64)`

Goal: Reach a target in 2D plane

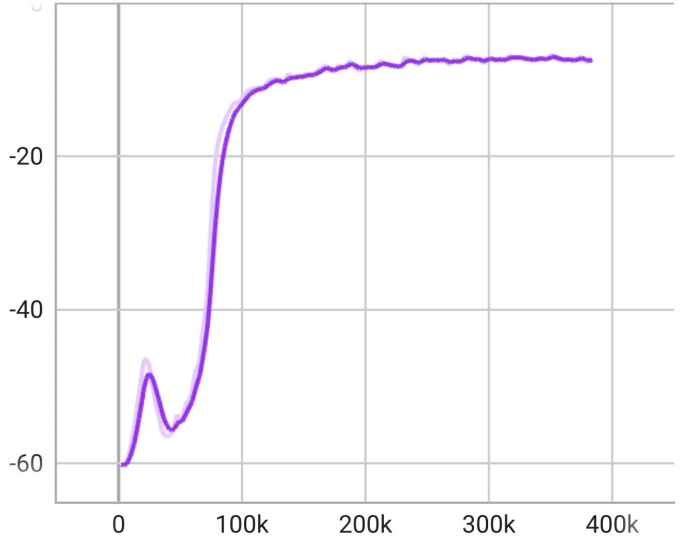
Reacher



MPO

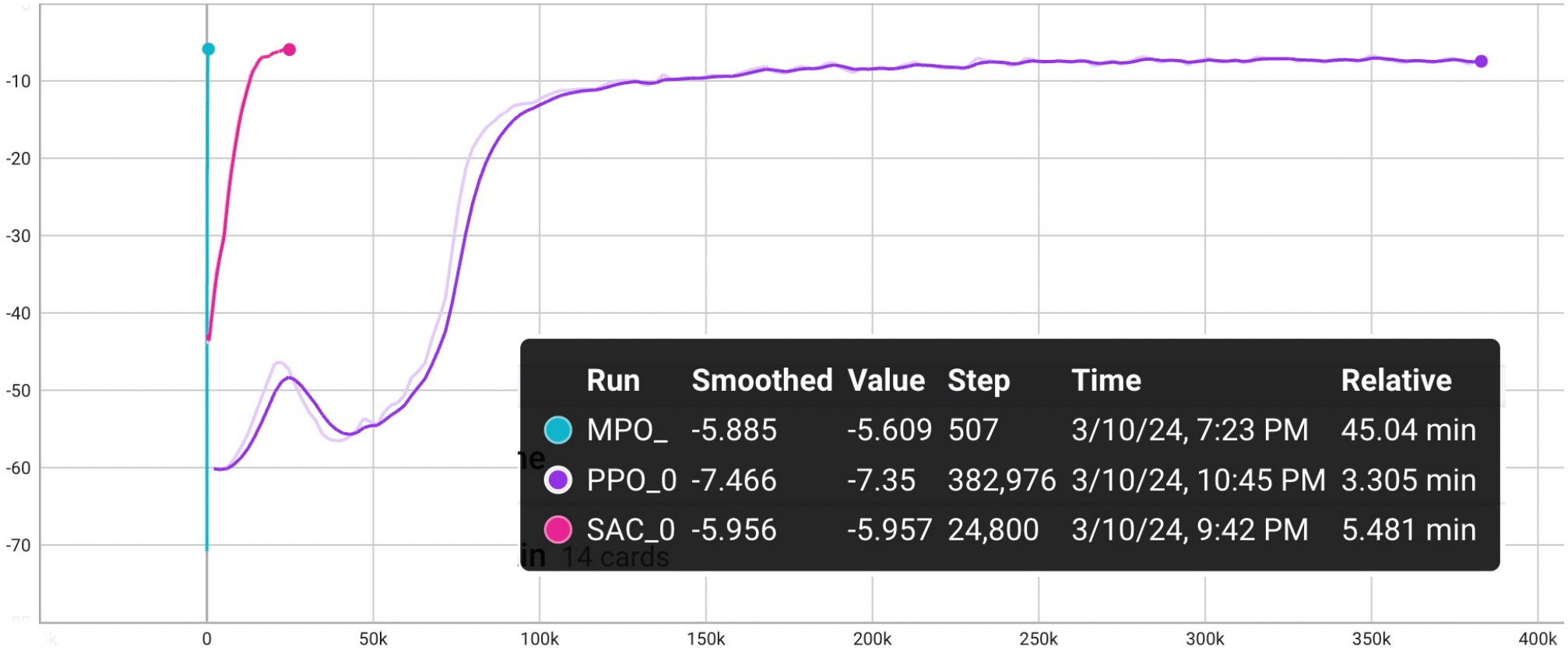


SAC



PPO

Reacher


-10
-20
-30
-40
-50
-60
-70
0
50k
100k
150k
200k
250k
300k
350k
400k
Run Smoothed Value Step Time Relative
MPO_ -5.885 -5.609 507 3/10/24, 7:23 PM 45.04 min
PPO_0 -7.466 -7.35 382,976 3/10/24, 10:45 PM 3.305 min
SAC_0 -5.956 -5.957 24,800 3/10/24, 9:42 PM 5.481 min

Hyperparameters

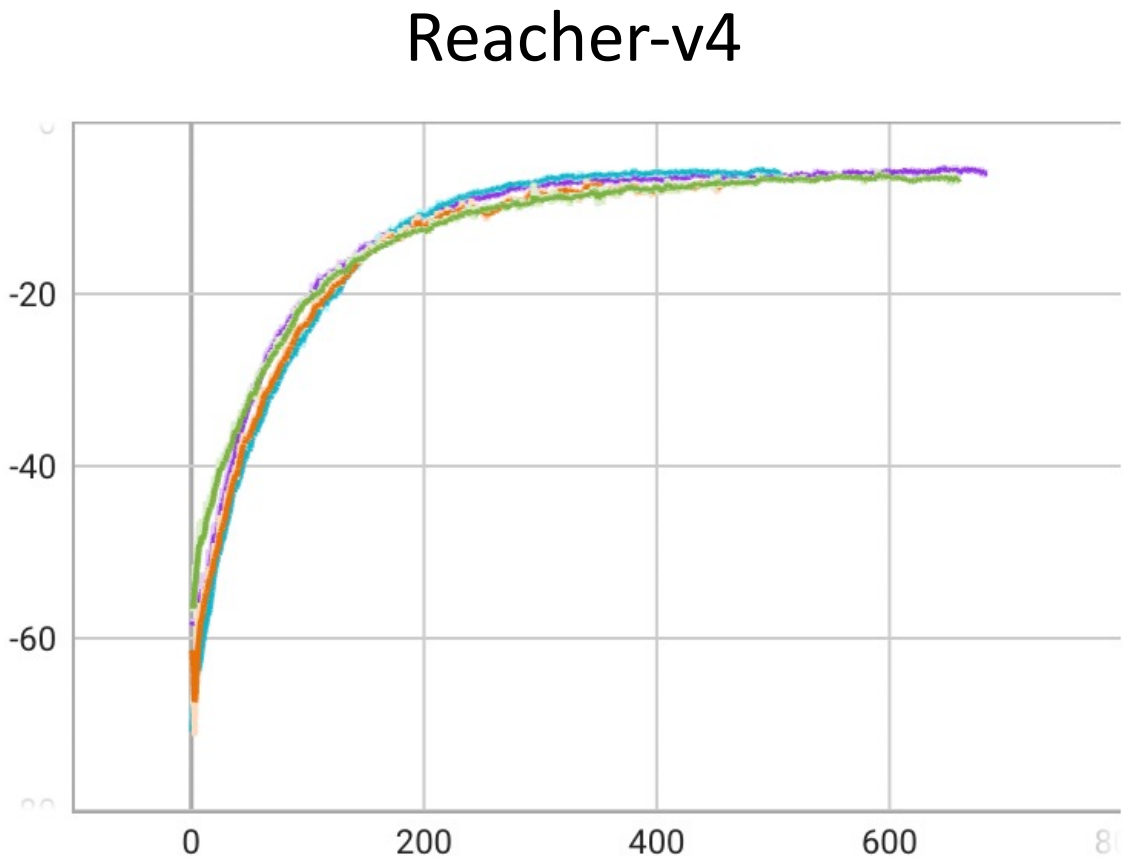
Parameter	Value
Policy network	100-100 Fully connected layers
Q-function network	200-200 Fully connected layers
Dual constraint ϵ	0.1
Mean constraint ϵ_μ	0.1
Covariance constraint ϵ_Σ	0.0001
Discount factor γ	0.99
Adam learning rate	0.0005

Where can we compromise?

Parameter	Value
Policy network	100-100 Fully connected layers
Q-function network	200-200 Fully connected layers
Dual constraint ϵ	0.1
Mean constraint ϵ_μ	0.1
Covariance constraint ϵ_Σ	0.0001
Discount factor γ	0.99
Adam learning rate	0.0005

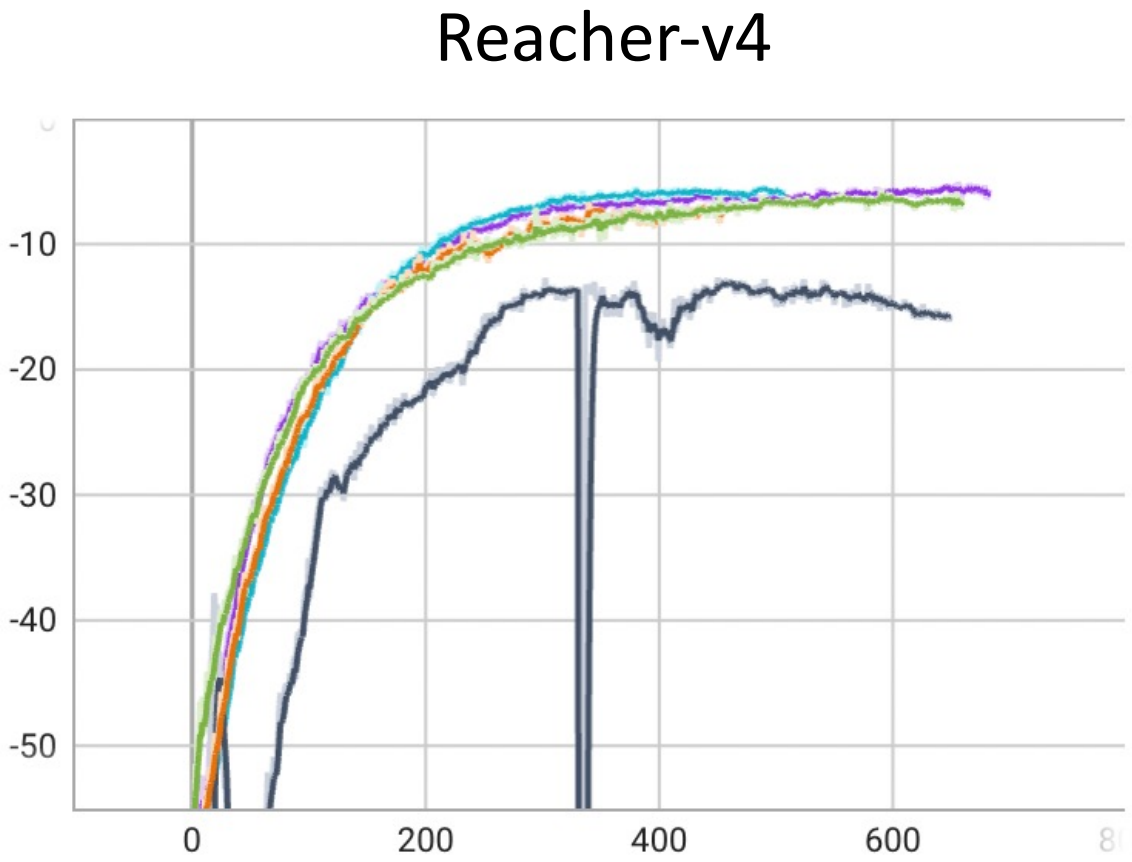
Changing network size

- MPO Paper:
  - Policy Network: 100-100
  - Q-Network: 200-200
- Smallest Size:
  - Policy Network: 25-25
  - Q-Network: 50-50
- Largest Size:
  - Policy Network: 256-256
  - Q-Network: 256-256

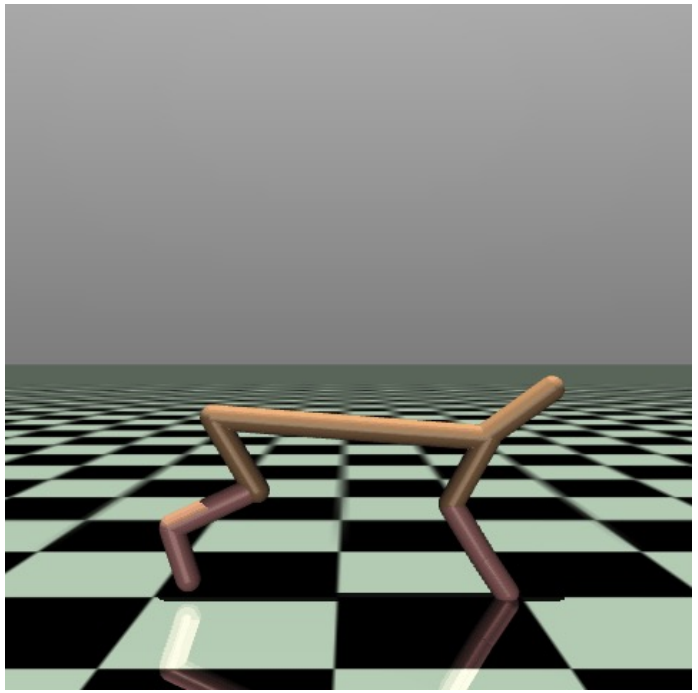


Changing network size

- MPO Paper:
  - Policy Network: 100-100
  - Q-Network: 200-200
- **Smallest Size:**
  - **Policy Network: 5-5**
  - **Q-Network: 5-5**
- Largest Size:
  - Policy Network: 256-256
  - Q-Network: 256-256



Cheetah



Action Space	`Box(-1.0, 1.0, (6,), float32)`
Observation Space	`Box(-inf, inf, (17,), float64)`

Goal: Move forward with minimal control effort

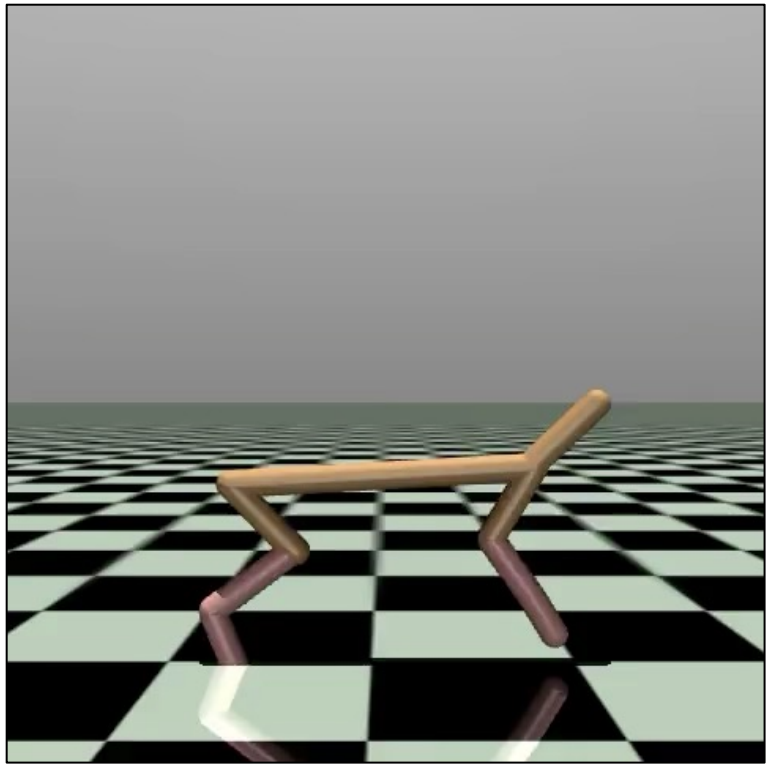
Where can we compromise?

Parameter	Value
Policy network	100-100 Fully connected layers
Q-function network	200-200 Fully connected layers
Dual constraint ϵ	0.1
Mean constraint ϵ_μ	0.1
Covariance constraint ϵ_Σ	0.0001
Discount factor γ	0.99
Adam learning rate	0.0005

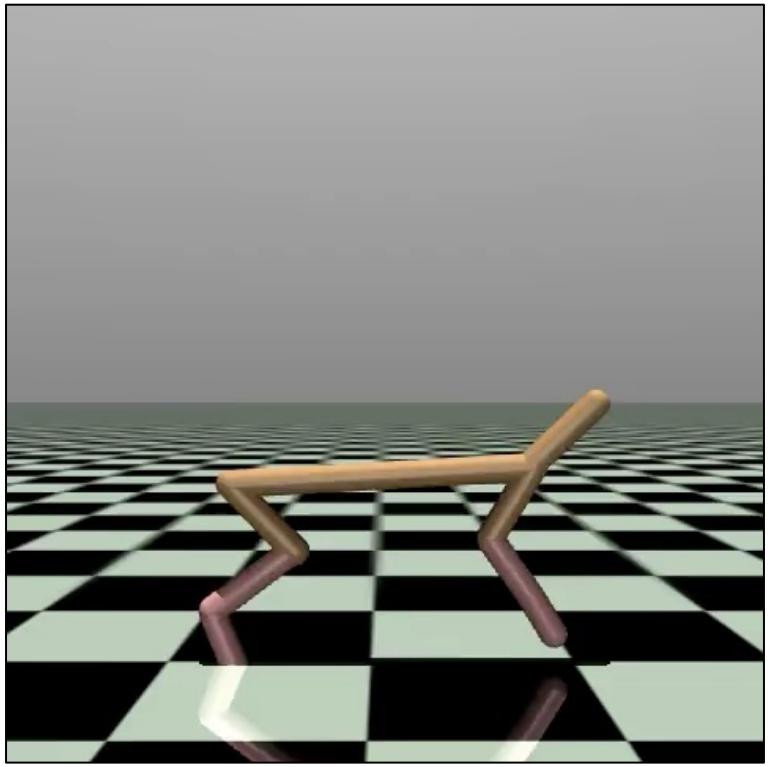
Where can we compromise?

Parameter	Value
Policy network	100-100 Fully connected layers
Q-function network	200-200 Fully connected layers
Dual constraint ϵ	0.1
Mean constraint ϵ_μ	0.1
Covariance constraint ϵ_Σ	0.0001
Discount factor γ	0.99
Adam learning rate	0.0005
Replay Buffer size	?
Samples from buffer	1000

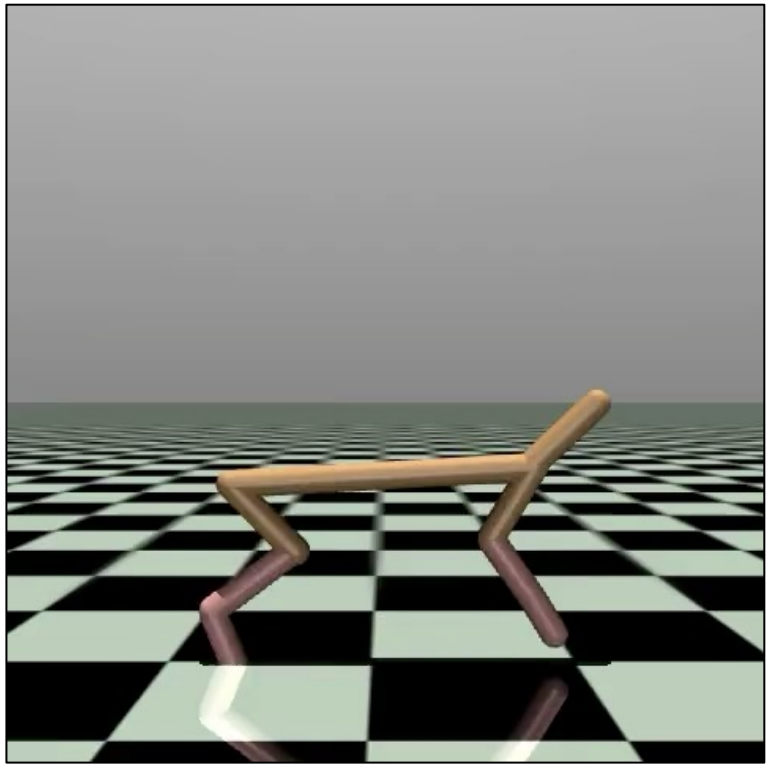
Cheetah



Iteration 50

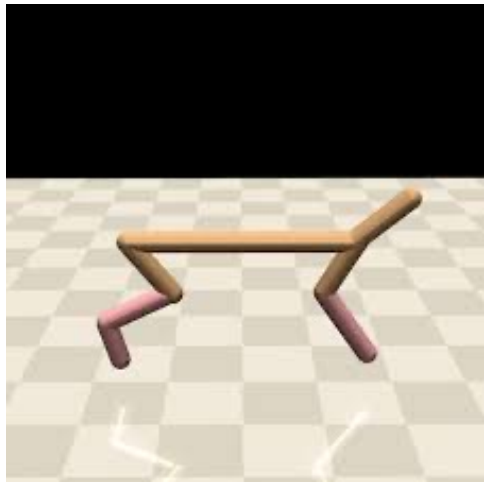


Iteration 100

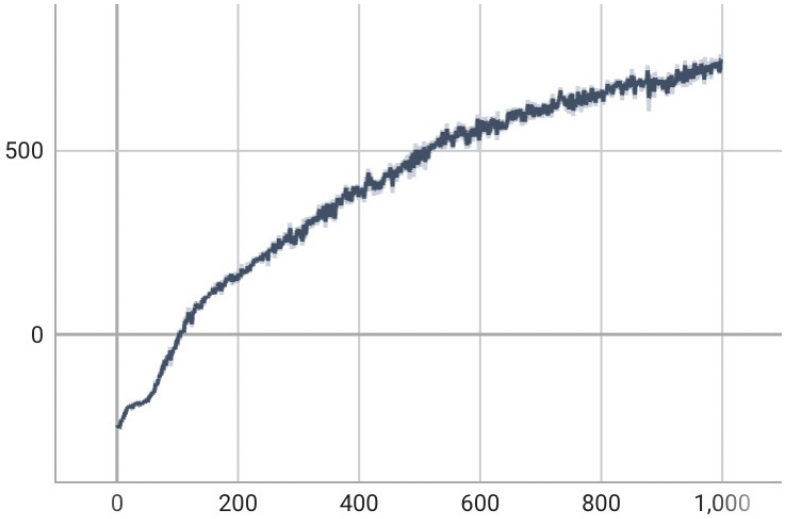


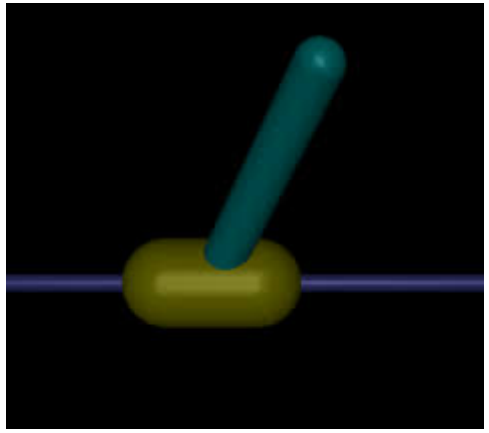
Iteration 1000

Cheetah vs Pendulum

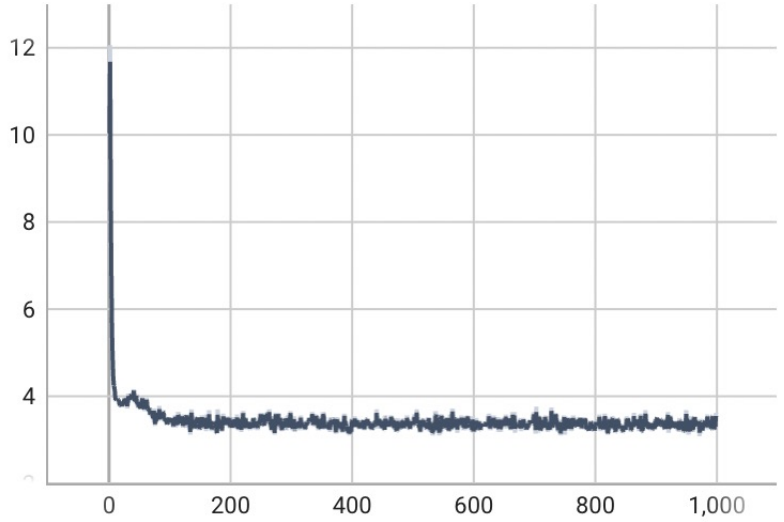


Cheetah

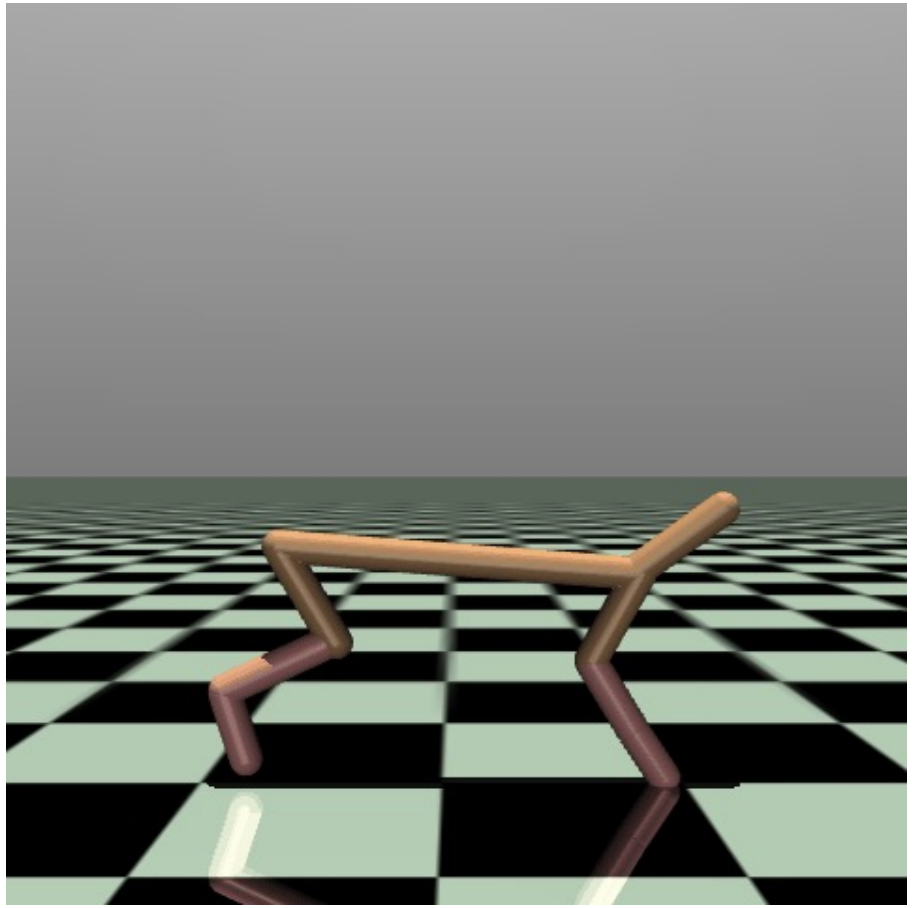




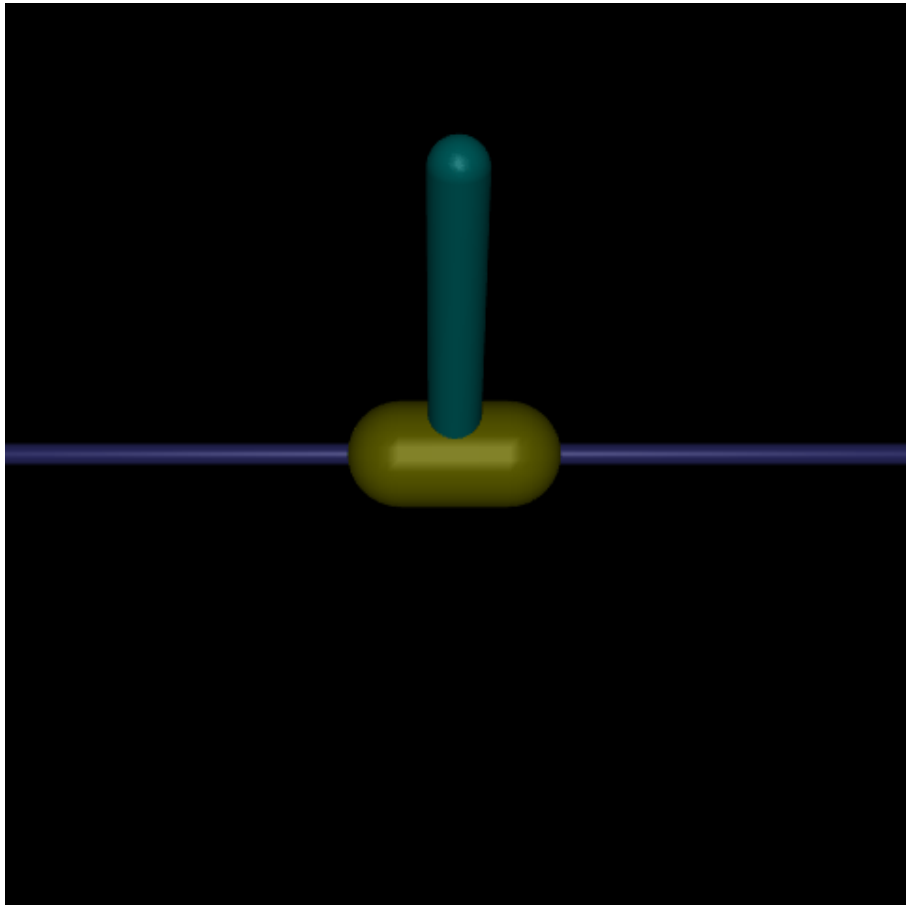
Inverted Pendulum



Cheetah vs Pendulum



Half Cheetah



Inverted Pendulum

Results

- Very high sampling efficiency
- Excellent end performance
- Highly robust to network size changes
- Robustness to buffer size change depends on termination condition

Limitations

1. Limited Exploration Mechanisms
2. Computationally complex
3. Implementation complexity
   - Optimization withing optimization
4. Dependence on Quality of Off-Policy Data
5. No official implementation available
   - Code optimizations

More to Learn

Parameter	Value
Policy network	100-100 Fully connected layers
Q-function network	200-200 Fully connected layers
Dual constraint ϵ	0.1
Mean constraint ϵ_μ	0.1
Covariance constraint ϵ_Σ	0.0001
Discount factor γ	0.99
Adam learning rate	0.0005

More to Learn

Parameter	Value
Policy network	100-100 Fully connected layers
Q-function network	200-200 Fully connected layers
Dual constraint ϵ	0.1
Mean constraint ϵ_μ	0.1
Covariance constraint ϵ_Σ	0.0001
Discount factor γ	0.99
Adam learning rate	0.0005

Additional Parameter
No. of episodes to sample
Length of each episode
Total episodes to sample
No. of actions to sample
Training batch size
No. of batches from the same buffer
Scaling factor of mean Lagrangian multiplier
Scaling factor of covariance Lagrangian multiplier
Constraint on mean Lagrangian multiplier
Constraint on mean Lagrangian multiplier
Optimization of M-step

References

- Textual References:
  - Maximum a posteriori Policy Optimisation (MPO)
  - Maximum a Posterior Policy Optimization - Gao Yue
  - The EM Algorithm Explained - Chloe Bi
  - Gymnasium - The Farama Foundation
  - Deep RL in Python
- Code References:
  - Stable Baselines 3
  - Project_RL - Theo Gruner
  - Replication MPO
  - Tonic - Fabio Pardo
  - ACME - Google DeepMind

Thank you.