
“The test of a man isn’t what you think he’ll do. It’s what he actually does.”
Frank Herbert, *Dune*
CMPUT 628
Deep RL
Marlos C. Machado
https://pbs.twimg.com/media/FGHKuYcXwAA31Zx.jpg
Class 18-19/ 25

Reminders & Notes

- Assignment 4 is due on March 14
- Midterm is on March 19
- Seminars will start the week after

Reminders & Notes

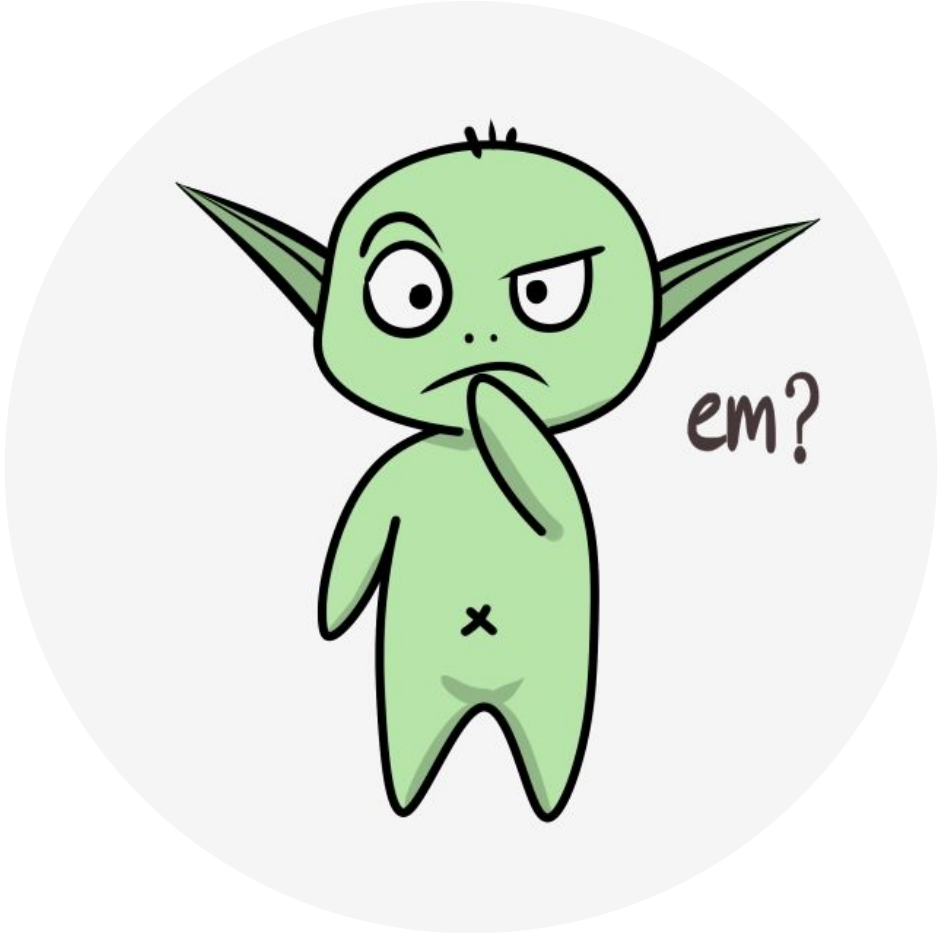
- Assignment 3 is marked
- Assignment 4 was due on March 14
- **Midterm is on March 24**
- Seminars will start the class after

Please, interrupt me at any time!

https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Last Class – DDPG, TD3, A2C, and A3C

- DDPG extends DQN to an actor-critic setting to allow for continuous actions
- DDPG is quite unstable though, so TD3 tries to stabilize DDPG by controlling overestimation bias, delaying policy updates, and regularizing the policy
- If we want stochastic policies, we can in fact sort of use traditional actor-critic methods; but using the advantage function is much more stable, allowing even for on-policy updates; thus A2C and A3C



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Max-Entropy Reinforcement Learning

- As I mentioned when discussing TD3, DDPG can be quite unstable
- What if we want to optimize a stochastic policy in an off-policy way?
- Instead of everything TD3 did, we can use the max-entropy framework, which reformulates the problem by adding entropy to the policy
  - More stable
  - Better exploration
  - Less overfitting

$$\max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t \Big(R_{t+1} + \beta \mathcal{H}\big(\pi(\cdot|S_t)\big) \Big) \right]$$

$$H(X) = -\sum_{x \in \mathbb{X}} p(x) \log p(x)$$

Entropy-Regularized Reinforcement Learning

- Entropy: a measure of disorder or randomness in a system
- In reinforcement learning, one is expected to balance between the actual return and the entropy of the policy

$$v_\pi(S) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t\Big(R(s_t,a_t,s_{t+1}) + \alpha H(\cdot|s_t)\Big)\Big|S_0 = s\right]$$

$$q_\pi(s,a) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t,s_{t+1}) + \alpha\sum_{t=1}^{\infty}\gamma^t H(\cdot|s_t)\Big|s_0 = s, a_o = a\right]$$

$$\pi(a) = \frac{1}{Z}\exp\left(\frac{1}{\beta}Q(s,a)\right) \qquad Z = \sum_a \exp\left(\frac{1}{\beta}Q(s,a)\right)$$

SAC [Haarnoja et al., 2018] is the maximum entropy version of DDPG

- Critic:

$$Y_{\text{SAC}} = R_{t+1} + \gamma \Big(\min \big(Q(S_{t+1}, \tilde{A}_{t+1}; \boldsymbol{\theta}_1^-), Q(S_{t+1}, \tilde{A}_{t+1}; \boldsymbol{\theta}_2^-) \big) - \beta \log \pi(\tilde{A}_{t+1} | S_{t+1}; \boldsymbol{\phi}) \Big)$$

$$\mathcal{L}^{\text{SAC}} = \mathbb{E}_{(o,a,r,o') \sim U(\mathcal{D})} \Big[\big(Y_{\text{SAC}} - Q(O_t, A_t; \boldsymbol{\theta}) \big)^2 \Big]$$

Clipped Double-Q Learning

Action doesn't come from the replay buffer, it is selected by the agent from $\pi(\cdot \mid S_{t+1}; \phi)$

- Actor:

$$\text{maximize} \Big[\min \big(Q(S_t, \tilde{A}_t; \boldsymbol{\theta}_1), Q(S_t, \tilde{A}_t; \boldsymbol{\theta}_2) \big) - \beta \log \pi \big(\tilde{A}(S_t; \boldsymbol{\phi}, \xi) | S_t; \boldsymbol{\phi} \big) \Big]$$

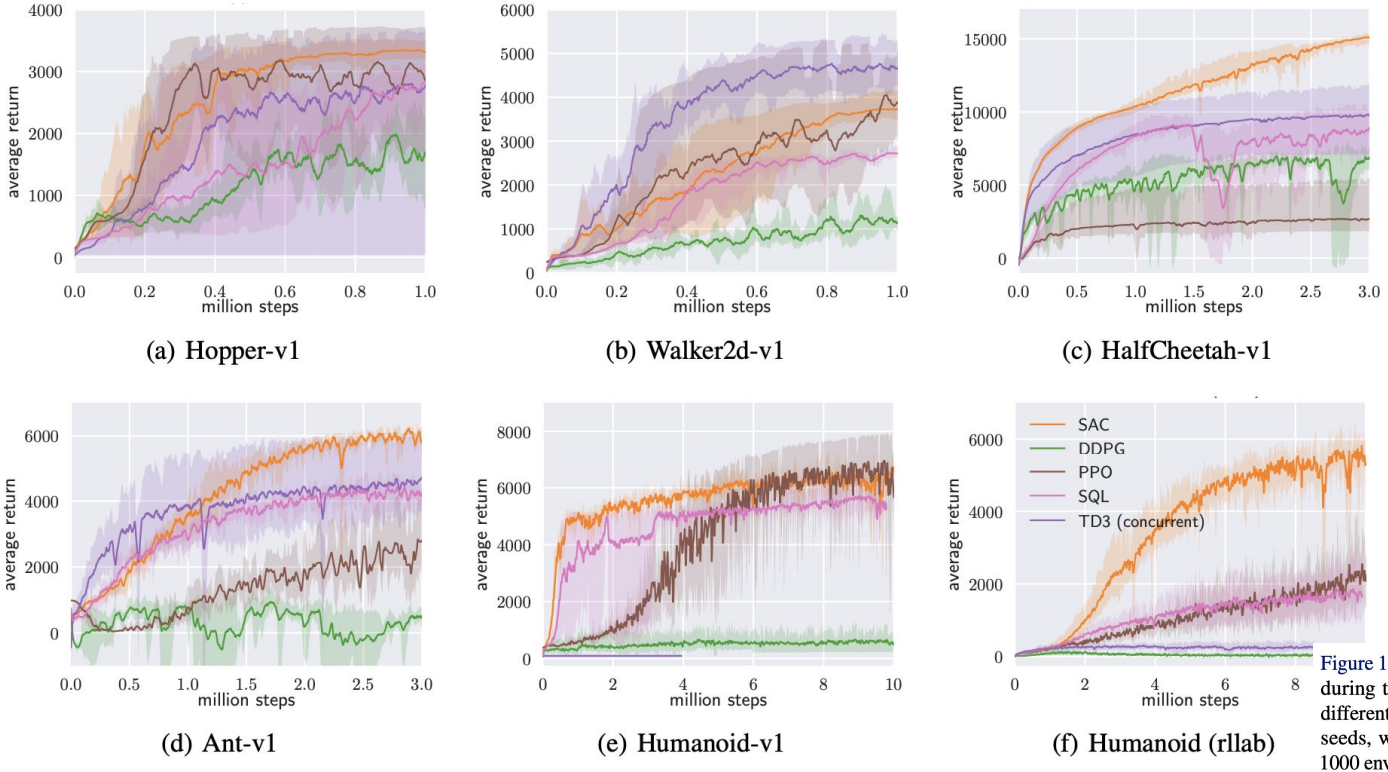
Reparameterization trick

$$\boldsymbol{\theta}_i^- \leftarrow \rho \boldsymbol{\theta}_i^- + (1 - \rho) \boldsymbol{\theta}_i$$

$$\tilde{A}(S_t; \boldsymbol{\phi}, \xi) = \tanh \big(\mu(S_t; \boldsymbol{\phi}) + \sigma(S_t; \boldsymbol{\phi} \odot \xi) \big)$$

$$\xi \sim \mathcal{N}(0, I)$$

SAC: Soft Actor-Critic [Haarnoja et al., 2018]

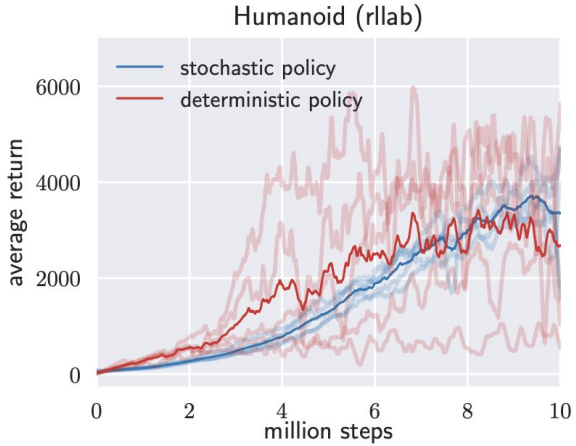


(a) Hopper-v1 (b) Walker2d-v1 (c) HalfCheetah-v1

(d) Ant-v1 (e) Humanoid-v1 (f) Humanoid (rllab)

Figure 1 shows the total average return of evaluation rollouts during training for DDPG, PPO, and TD3. We train five different instances of each algorithm with different random seeds, with each performing one evaluation rollout every 1000 environment steps. The solid curves corresponds to the mean and the shaded region to the minimum and maximum returns over the five trials.

The benefits of stochastic policies for training and of deterministic policies for evaluation [Haarnoja et al., 2018]



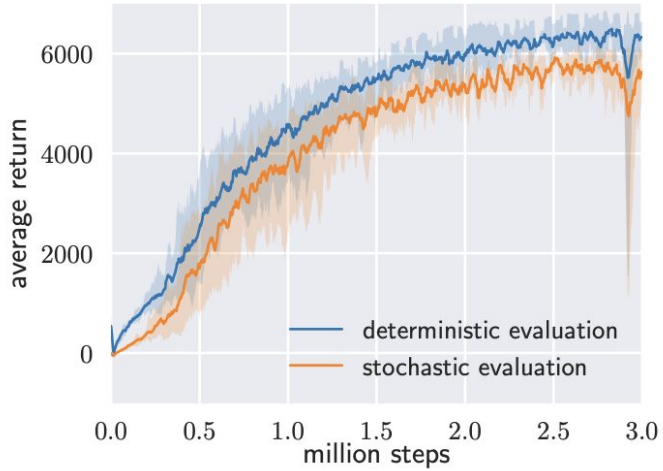
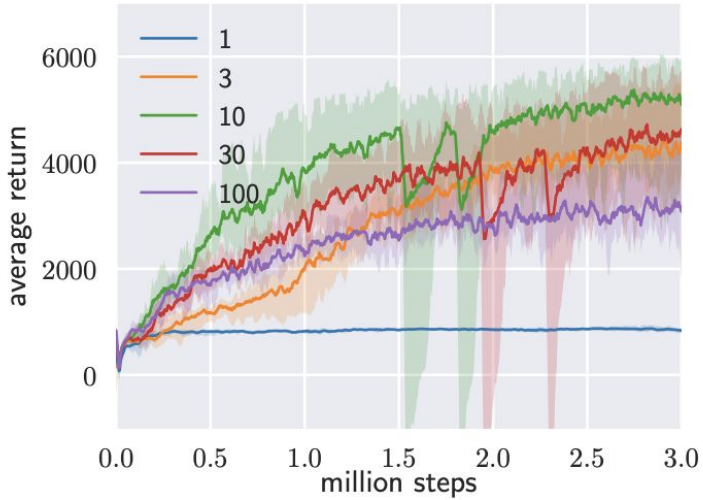
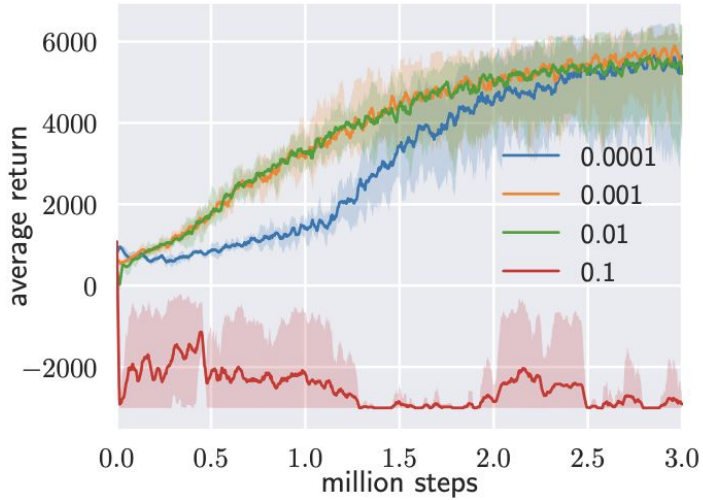


Figure 2. Comparison of SAC (blue) and a deterministic variant of SAC (red) in terms of the stability of individual random seeds on the Humanoid (rllab) benchmark. The comparison indicates that stochasticity can stabilize training as the variability between the seeds becomes much higher with a deterministic policy.

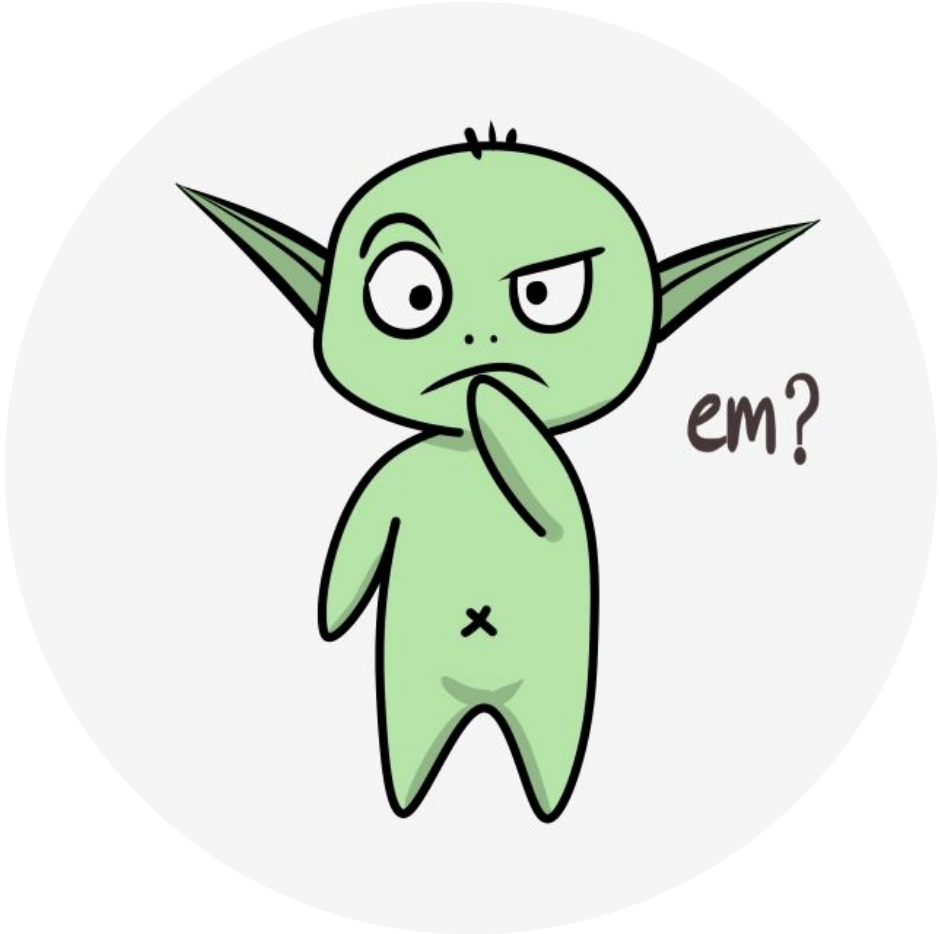
Some choices do matter in SAC [Haarnoja et al., 2018]



(b) Reward Scale



(c) Target Smoothing Coefficient (τ)



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

PG methods can be unstable, and the stable ones are slow

- Policy gradient methods take a gradient ascent step in the direction of maximum expected return; but that is estimated through *samples*
- Because these are sample-based methods, we might end up being misled and take steps in the wrong direction, increasing the likelihood of taking suboptimal actions
- By taking suboptimal actions, we change the data distribution by spending more time in states we wouldn't otherwise spend time on
- The poor data distribution can lead to even poorer updates, eventually leading to a complete collapse in performance

TRPO: Trust Region Policy Optimization [Schulman et al., 2015]

- The general idea behind TRPO is to take the biggest possible step to improve policy performance while ensuring that the new policy is not going to be too different from the old policy
- To have a better behaved optimization, and decide on the size of the step we will take, we can look at the curvature of the space (not only the gradient at a point)

Gradient descent: Line search, decide on direction and then take a step on that direction.

Trust region: We decide on the maximum step we want to take and then we find the optimal point within these constraints.

Re-deriving the PG theorem using importance sampling

- Based on Pieter Abbeel's L4 TRPO and PPO (Foundations of Deep RL Series)
  https://www.youtube.com/watch?v=KjWF8VIMGiY

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\tau \sim \boldsymbol{\theta}_{\text{old}}} \left[\frac{P(\tau|\boldsymbol{\theta})}{P(\tau|\boldsymbol{\theta}_{\text{old}})} R(\tau) \right] \qquad \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\tau \sim \boldsymbol{\theta}_{\text{old}}} \left[\frac{\nabla_{\boldsymbol{\theta}} P(\tau|\boldsymbol{\theta})}{P(\tau|\boldsymbol{\theta}_{\text{old}})} R(\tau) \right]$$

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{\text{old}}} = \mathbb{E}_{\tau \sim \boldsymbol{\theta}_{\text{old}}} \left[\frac{\nabla_{\boldsymbol{\theta}} P(\tau|\boldsymbol{\theta}) \big|_{\boldsymbol{\theta}_{\text{old}}}}{P(\tau|\boldsymbol{\theta}_{\text{old}})} R(\tau) \right]$$

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{\text{old}}} = \mathbb{E}_{\tau \sim \boldsymbol{\theta}_{\text{old}}} \left[\nabla_{\boldsymbol{\theta}} \log P(\tau|\boldsymbol{\theta}) \big|_{\boldsymbol{\theta}_{\text{old}}} R(\tau) \right]$$

TRPO: Trust Region Policy Optimization [Schulman et al., 2015]

- We want to make sure the new policy is not too different from the old policy

$$\max_{\pi} \mathcal{L}(\pi) = \mathbb{E}_{\pi_{\text{old}}} \left[\frac{\pi(a|s)}{\pi_{\text{old}}(a|s)} A_{\pi_{\text{old}}}(s,a) \right]$$

$$\text{s.t.} \quad \mathbb{E}_{\pi_{\text{old}}} \left[KL(\pi || \pi_{\text{old}}) \right] \leq \epsilon$$

$$KL(P||Q) = \sum_{x \in \mathcal{X}} P(x) \log \frac{P(x)}{Q(x)}$$

How do we solve this optimization problem, which has constraints?

TRPO: Trust Region Policy Optimization [Schulman et al., 2015]

- We can leverage some aspects of *natural policy gradient* to solve this objective

$$\pi_{k+1} = \arg\max_{\pi} \mathcal{L}_{\pi_k}(\pi) \qquad \text{s.t.} \quad KL(\pi||\pi_k) \leq \delta$$

$$\mathcal{L}_{\boldsymbol{\theta}_k}(\boldsymbol{\theta}) \approx \cancel{\mathcal{L}_{\boldsymbol{\theta}_k}(\boldsymbol{\theta}_k)} + \boldsymbol{g}^\top(\boldsymbol{\theta} - \boldsymbol{\theta}_k) + \ \ldots$$

$$KL(\boldsymbol{\theta}||\boldsymbol{\theta}_k) \approx \cancel{KL(\boldsymbol{\theta}_k||\boldsymbol{\theta}_k)} + \cancel{\nabla_{\boldsymbol{\theta}} KL(\boldsymbol{\theta}||\boldsymbol{\theta}_k)|_{\boldsymbol{\theta}_k}}(\boldsymbol{\theta} - \boldsymbol{\theta}_k) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_k)^\top \boldsymbol{H}(\boldsymbol{\theta} - \boldsymbol{\theta}_k)$$

TRPO: Trust Region Policy Optimization [Schulman et al., 2015]

- We can leverage some aspects of *natural policy gradient* to solve this objective

$$\pi_{k+1} = \arg\max_{\pi} \mathcal{L}_{\pi_k}(\pi) \qquad \text{s.t.} \quad KL(\pi||\pi_k) \leq \delta$$

$$\mathcal{L}_{\boldsymbol{\theta}_k}(\boldsymbol{\theta}) \approx \boldsymbol{g}^\top(\boldsymbol{\theta} - \boldsymbol{\theta}_k)$$

$$KL(\boldsymbol{\theta}||\boldsymbol{\theta}_k) \approx \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_k)^\top \boldsymbol{H}(\boldsymbol{\theta} - \boldsymbol{\theta}_k)$$

TRPO: Trust Region Policy Optimization [Schulman et al., 2015]

- We can leverage some aspects of *natural policy gradient* to solve this objective

$$\boldsymbol{\theta}_{k+1} = \arg\max_{\boldsymbol{\theta}} \boldsymbol{g}^\top (\boldsymbol{\theta} - \boldsymbol{\theta}_k) \qquad \text{s.t.} \quad \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_k)^\top \boldsymbol{H} (\boldsymbol{\theta} - \boldsymbol{\theta}_k) \leq \epsilon$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \sqrt{\frac{2\epsilon}{\boldsymbol{g}^\top \boldsymbol{H}^{-1} \boldsymbol{g}}} \boldsymbol{H}^{-1} \boldsymbol{g}$$

We can use the conjugate gradient to directly compute these. See Schulman et al. (2015)
Or Jonathan Hui's blog post [link]

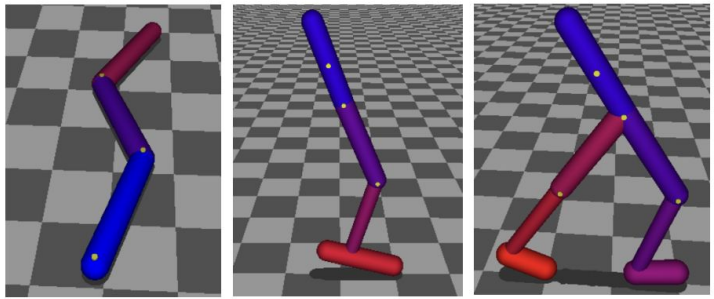
TRPO: Trust Region Policy Optimization [Schulman et al., 2015]

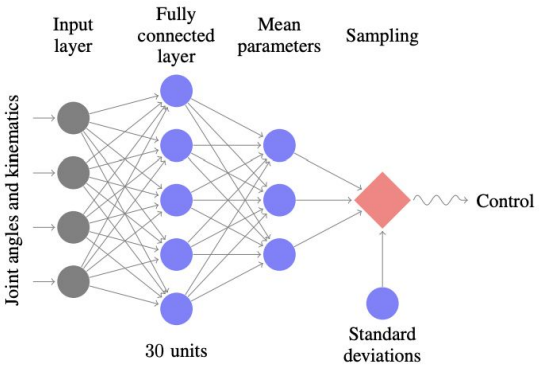
1. Collect *trajectories* D_k *by running policy* $\pi_k = \pi(\boldsymbol{\theta}_k)$ *in the environment*
2. *Estimate Advantages* A_t *based on the current value function,* $V_{\phi k}$
3. *Estimate policy gradient as* $\hat{\boldsymbol{g}}_k = \frac{1}{\mathcal{D}_k} \sum_{\tau \in |\mathcal{D}_k|} \sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t)\big|_{\boldsymbol{\theta}_k} \hat{A}_t$
4. *Use the conjugate gradient algorithm to compute* $\hat{\boldsymbol{x}}_k \approx \hat{\boldsymbol{H}}^{-1} \hat{\boldsymbol{g}}_k$
5. *Update policy by backtracking line search with exponential decay*

   $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha^j \sqrt{\frac{2\epsilon}{\hat{\boldsymbol{x}}_k{}^\top \hat{\boldsymbol{H}}_k{}^{-1} \hat{\boldsymbol{x}}_k}} \hat{\boldsymbol{x}}_k$ *, where we sweep j until it satisfies the KL constraint*
6. *Update weights for the current value function*

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(S_t) - \hat{G}_t \right)^2$$

TRPO: Trust Region Policy Optimization [Schulman et al., 2015]





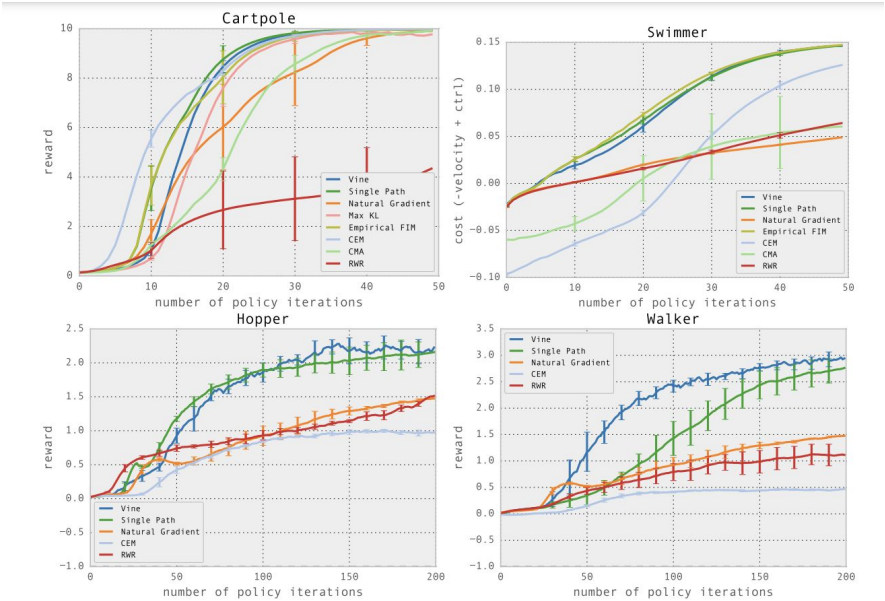
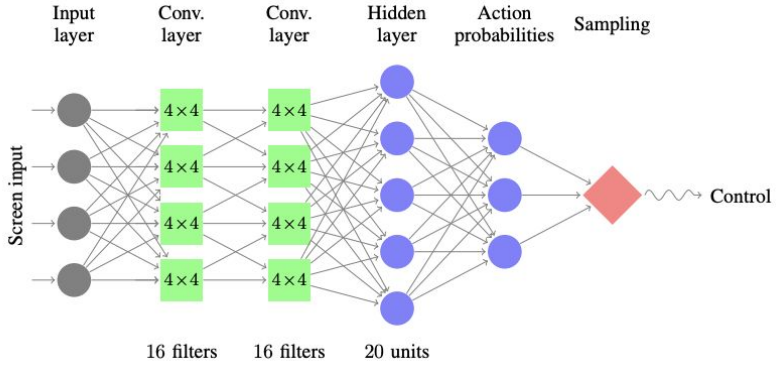
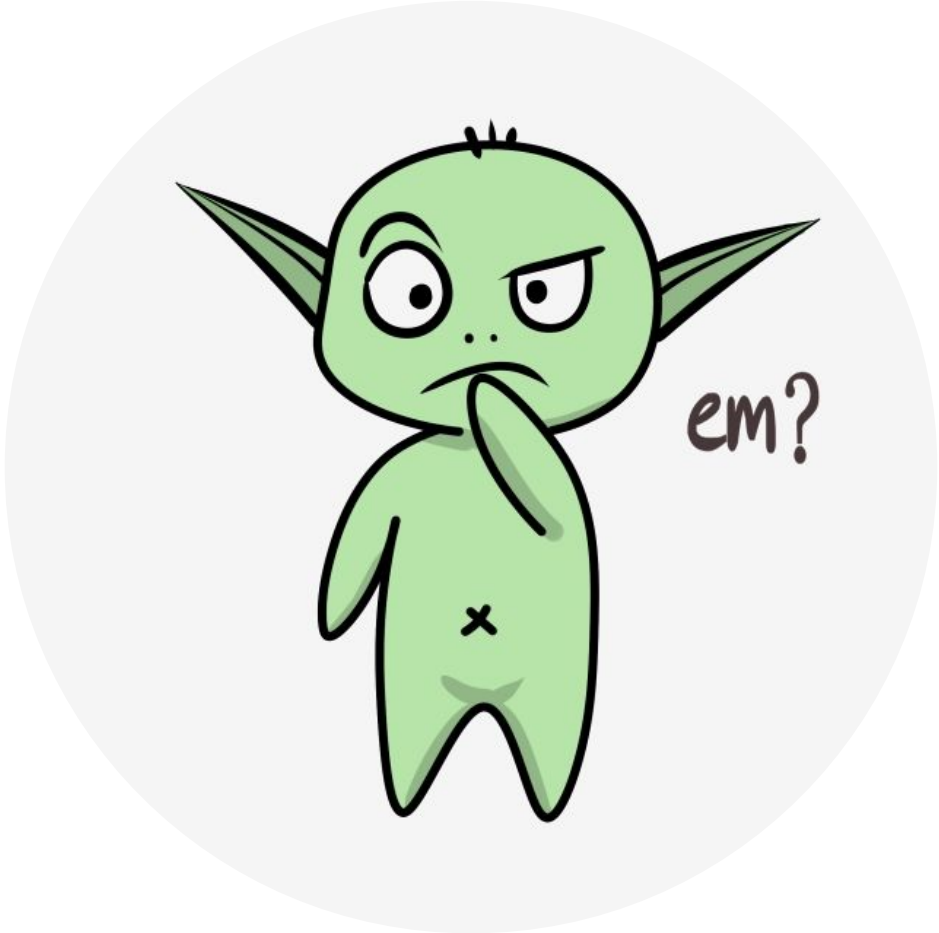


Figure 4. Learning curves for locomotion tasks, averaged across five runs of each algorithm with random initializations. Note that for the hopper and walker, a score of -1 is achievable without any forward velocity, indicating a policy that simply learned balanced standing, but not walking.

TRPO: Trust Region Policy Optimization [Schulman et al., 2015]



	B. Rider	*Breakout*	*Enduro*	*Pong*	*Q*bert*	*Seaquest*	*S. Invaders*
Random	354	1.2	0	−20.4	157	110	179
Human (Mnih et al., 2013)	7456	31.0	368	−3.0	18900	28010	3690
Deep Q Learning (Mnih et al., 2013)	4092	168.0	470	20.0	1952	1705	581
UCC-I (Guo et al., 2014)	5702	380	741	21	20025	2995	692
TRPO - single path	1425.2	10.8	534.6	20.9	1973.5	1908.6	568.4
TRPO - vine	859.5	34.2	430.8	20.9	7732.5	788.4	450.2



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

PPO: Proximal Policy Optimization [Schulman et al., 2017]

- For better scalability and simplicity, TRPO as a first order method?
  - It would also allow for shared parameters, aux. tasks, leveraging modern optimizers, and more…

TRPO:

$$\max_{\pi} \mathcal{L}(\pi) = \mathbb{E}_{\pi_{\text{old}}}\left[\frac{\pi(a|s)}{\pi_{\text{old}}(a|s)} A_{\pi_{\text{old}}}(s,a)\right]$$

$$\text{s.t.} \quad \mathbb{E}_{\pi_{\text{old}}}\Big[KL(\pi(\cdot|S_t)||\pi_{\text{old}}(\cdot|S_t))\Big] \leq \epsilon$$

PPO v1:

$$\mathbb{E}_{\pi_{\text{old}}}\left[\frac{\pi(A_t|S_t)}{\pi_{\text{old}}(A_t|S_t)} A_{\pi_{\text{old}}}(s,a)\right]$$

$$-\beta\Big(\mathbb{E}_{\pi_{\text{old}}}\Big[KL\big(\pi(\cdot|S_t)||\pi_{\text{old}}(\cdot|S_t)\big)\Big] - \epsilon\Big)$$

We should dynamically adjust β so it is bigger when the KL is large

PPO: Proximal Policy Optimization [Schulman et al., 2017]

- We can further simplify PPO v1

PPO v1:

$z(\theta)$

$$\mathbb{E}_{\pi_{\text{old}}}\left[\frac{\pi(A_t|S_t)}{\pi_{\text{old}}(A_t|S_t)}A_{\pi_{\text{old}}}(s,a)\right]$$

$$\mathbb{E}_{\pi_{\text{old}}}\left[\min\left(z(\theta)A_{\pi_{\text{old}}}(s,a), \left[z(\theta)\right]_{1-c}^{1+c}A_{\pi_{\text{old}}}(s,a)\right)\right]$$

$$-\beta\left(\mathbb{E}_{\pi_{\text{old}}}\left[KL\left(\pi(\cdot|S_t)||\pi_{\text{old}}(\cdot|S_t)\right)\right]-\epsilon\right)$$

Pessimistic bound: The farther you can go in terms of the ratio is 1 ± c

PPO: Proximal Policy Optimization [Schulman et al., 2017]

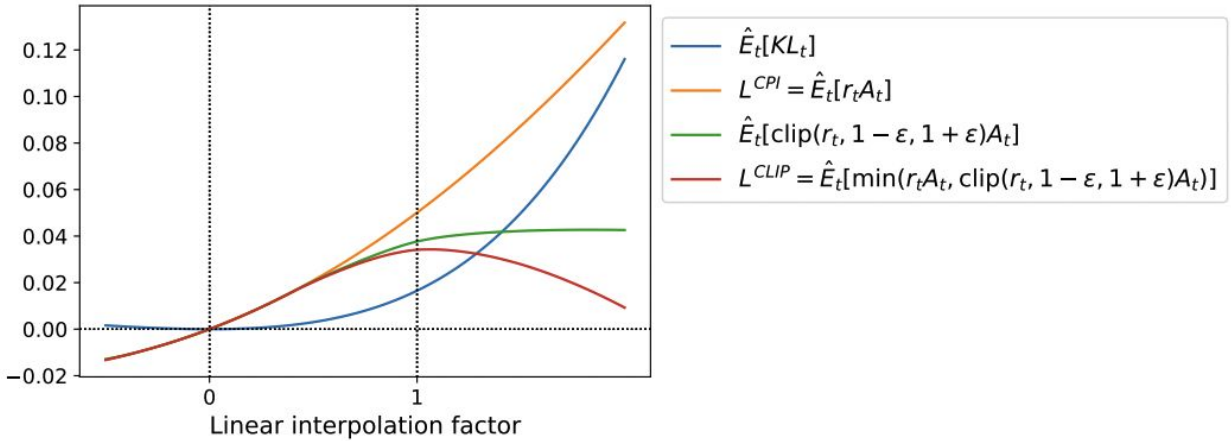
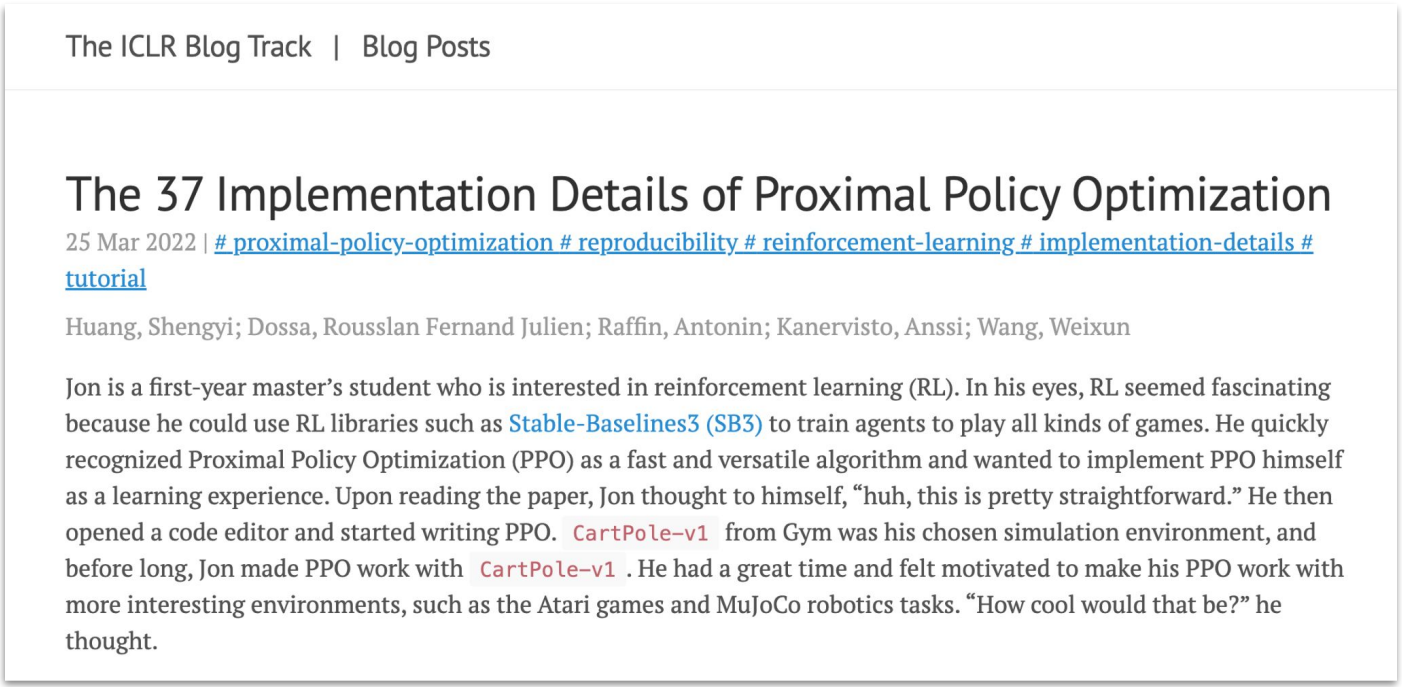


Figure 2: Surrogate objectives, as we interpolate between the initial policy parameter θ_{old}, and the updated policy parameter, which we compute after one iteration of PPO. The updated policy has a KL divergence of about 0.02 from the initial policy, and this is the point at which L^{CLIP} is maximal. This plot corresponds to the first policy update on the Hopper-v1 problem, using hyperparameters provided in Section 6.1.

PPO (and the others) have many implementation details



The ICLR Blog Track | Blog Posts

The 37 Implementation Details of Proximal Policy Optimization

25 Mar 2022 | # proximal-policy-optimization # reproducibility # reinforcement-learning # implementation-details # tutorial

Huang, Shengyi; Dossa, Rousslan Fernand Julien; Raffin, Antonin; Kanervisto, Anssi; Wang, Weixun

Jon is a first-year master's student who is interested in reinforcement learning (RL). In his eyes, RL seemed fascinating because he could use RL libraries such as Stable-Baselines3 (SB3) to train agents to play all kinds of games. He quickly recognized Proximal Policy Optimization (PPO) as a fast and versatile algorithm and wanted to implement PPO himself as a learning experience. Upon reading the paper, Jon thought to himself, "huh, this is pretty straightforward." He then opened a code editor and started writing PPO. `CartPole-v1` from Gym was his chosen simulation environment, and before long, Jon made PPO work with `CartPole-v1` . He had a great time and felt motivated to make his PPO work with more interesting environments, such as the Atari games and MuJoCo robotics tasks. "How cool would that be?" he thought.

A big picture view

- Methods like DDPG and SAC are sample efficient because they are off-policy
- Off-policy methods are unstable, though
- TRPO is on-policy, and quite stable, but complicated and it is hard to scale
- A2C and PPO scale amazingly well, and they are on-policy (thus, stable)
  - PPO successes include OpenAI Five, OpenAI solving a Rubik's Cube with a robot hand, & ChatGPT
- We just need to have some clarity over computation vs sample efficiency
  - Sometimes, interacting with the world is "expensive", and we want to squeeze as much as we can from samples, even if that is computationally demanding (e.g., SAC, DDPG)
  - Sometimes interacting with the world is "cheap", so we can afford to be sample inefficient and make the algorithm the bottleneck—oftentimes we can get stability out of this setting (e.g., PPO)

PPO is probably the most used RL algorithm, so it works ¯_(ツ)_/¯ [Schulman et al., 2017]

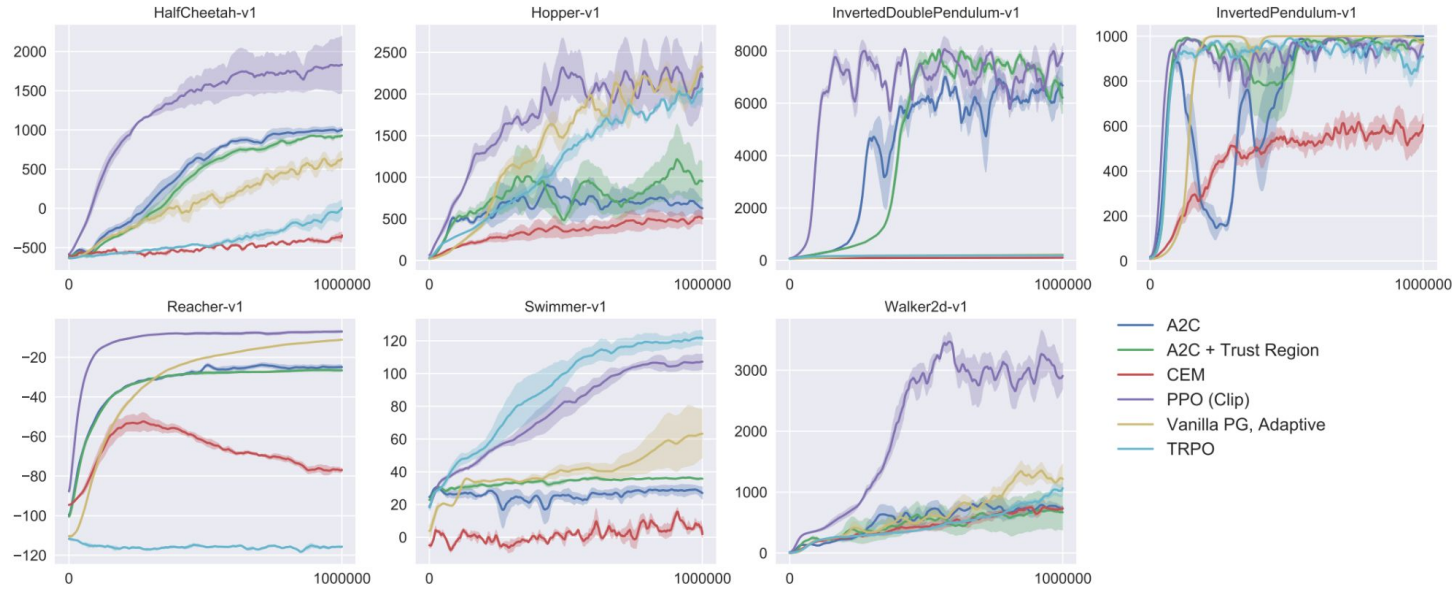
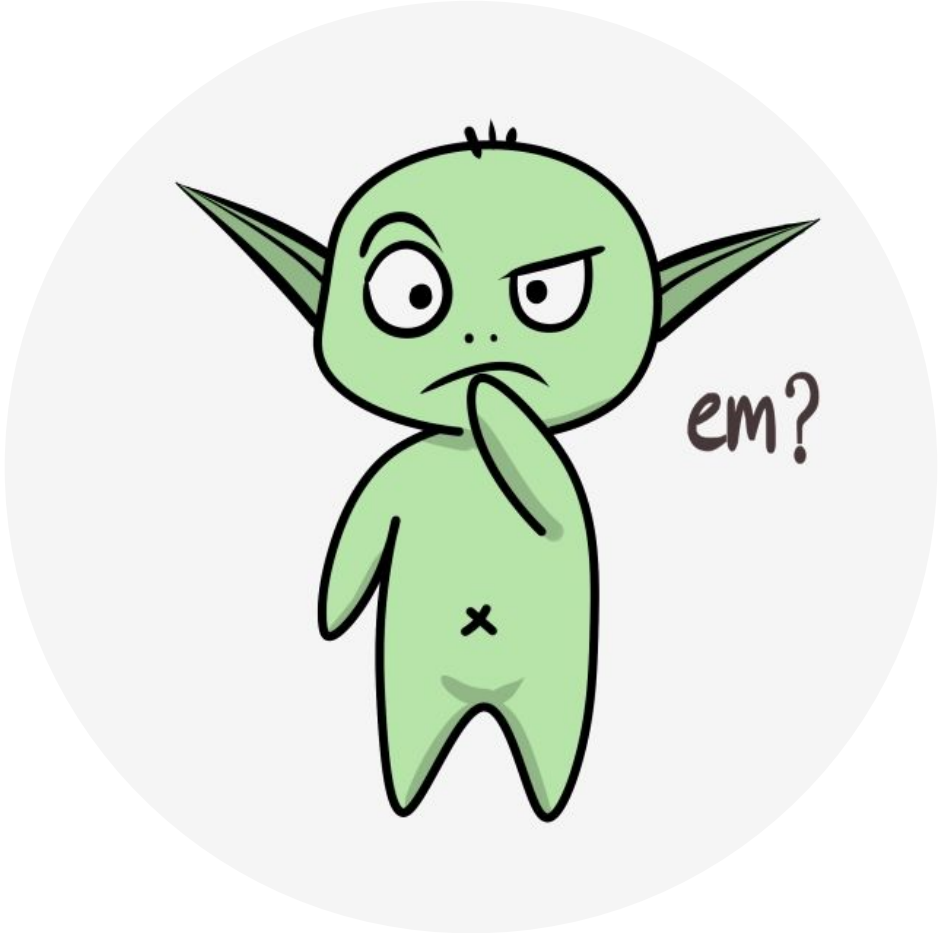


Figure 3: Comparison of several algorithms on several MuJoCo environments, training for one million timesteps.

PPO is probably the most used RL algorithm, so it works ¯_(ツ)_/¯ [Schulman et al., 2017]

	A2C	ACER	PPO	Tie
(1) avg. episode reward over all of training	1	18	**30**	0
(2) avg. episode reward over last 100 episodes	1	**28**	19	1

Table 2: Number of games "won" by each algorithm, where the scoring metric is averaged across three trials.



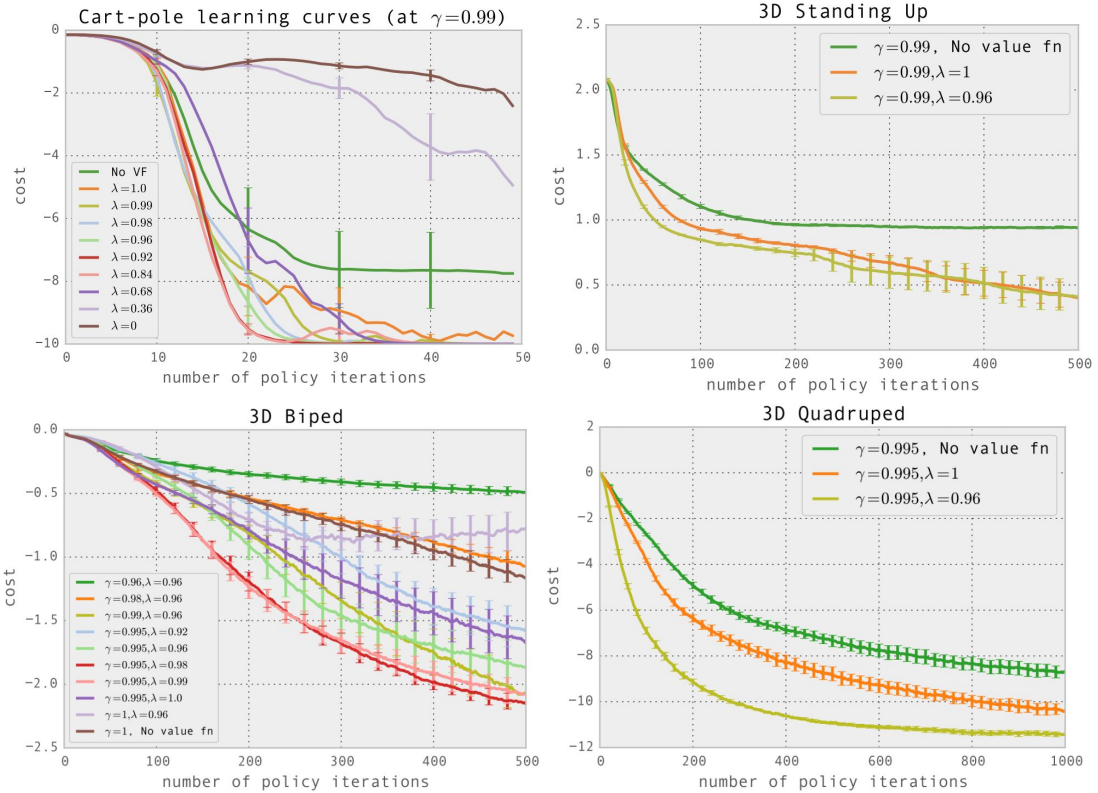
https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

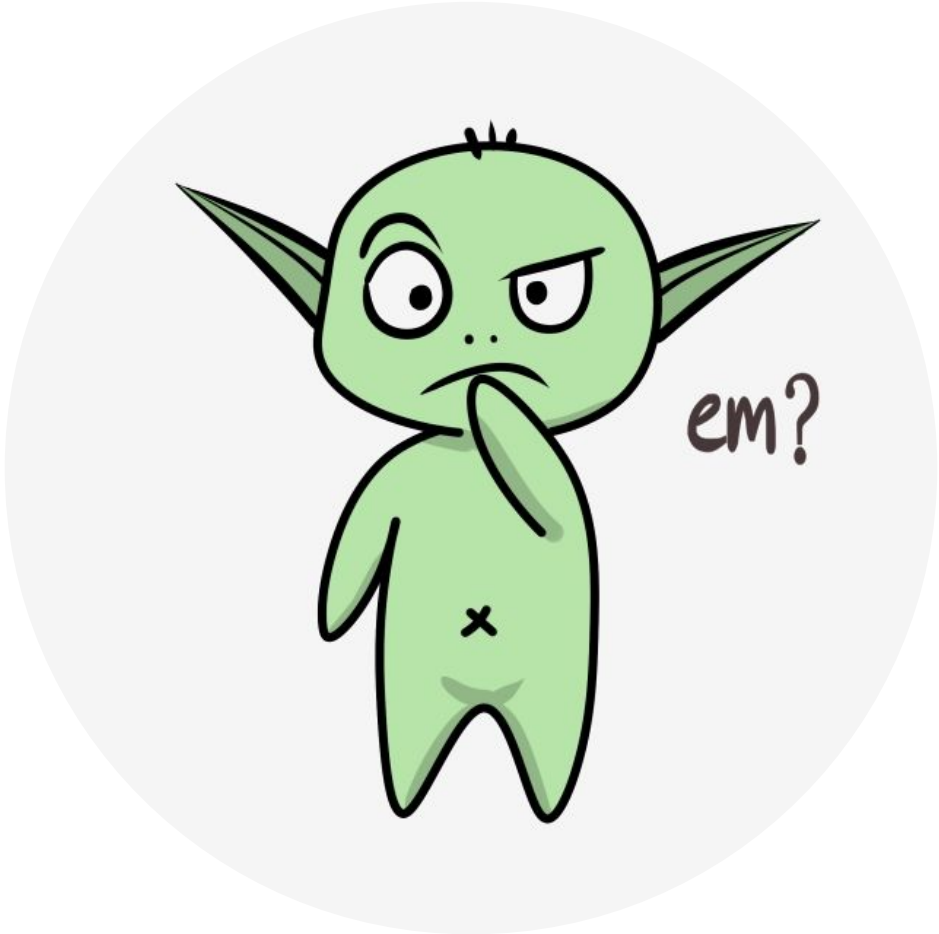
GAE: Generalized Advantage Estimation [Schulman et al., 2016]

- λ-returns were originally defined as the exponential average of all n-step returns
- GAE was defined by Schulman et al. (2016) as the exponential average of the advantage functions
  - Notice “that is analogous to TD(λ)” and “the formula has been proposed in prior work (Kimura & Kobayashi, 1998; Wawrzynski, 2009)”

$$
\begin{aligned}
\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} &:= (1-\lambda)\left(\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + \dots\right) \\
&= (1-\lambda)\left(\delta_t^V + \lambda(\delta_t^V + \gamma\delta_{t+1}^V) + \lambda^2(\delta_t^V + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+2}^V) + \dots\right) \\
&= (1-\lambda)(\delta_t^V(1 + \lambda + \lambda^2 + \dots) + \gamma\delta_{t+1}^V(\lambda + \lambda^2 + \lambda^3 + \dots) \\
&\qquad + \gamma^2\delta_{t+2}^V(\lambda^2 + \lambda^3 + \lambda^4 + \dots) + \dots) \\
&= (1-\lambda)\left(\delta_t^V\left(\frac{1}{1-\lambda}\right) + \gamma\delta_{t+1}^V\left(\frac{\lambda}{1-\lambda}\right) + \gamma^2\delta_{t+2}^V\left(\frac{\lambda^2}{1-\lambda}\right) + \dots\right) \\
&= \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V \qquad (16)
\end{aligned}
$$

GAE: Generalized Advantage Estimation helps [Schulman et al., 2016]





https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Next class

- MBRL (starting later), then Midterm
- What I recommend you to do for next class:
  - Read
    - J. Schrittwieser et al.: Mastering Atari, Go, chess and shogi by planning with a learned model. Nature 588(7839): 604-609 (2020)