LECTURE 3: POLICY GRADIENTS | PPO

Marius-Constantin Dinu

Reinforcement Learning Bootcamp, 25-27.09.2024

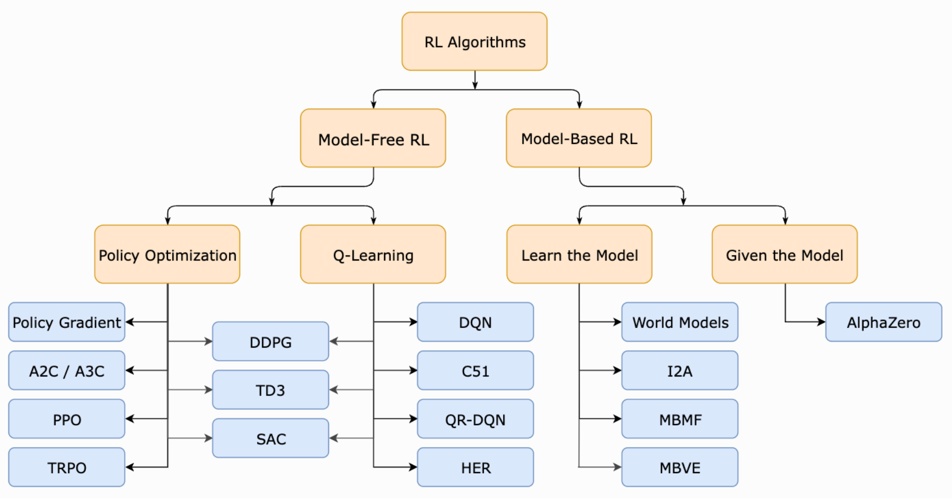
marius@extensity.ai

Johannes Kepler University

AI Austria | RL Community

ExtensityAI

RL Algorithms



PPO Results

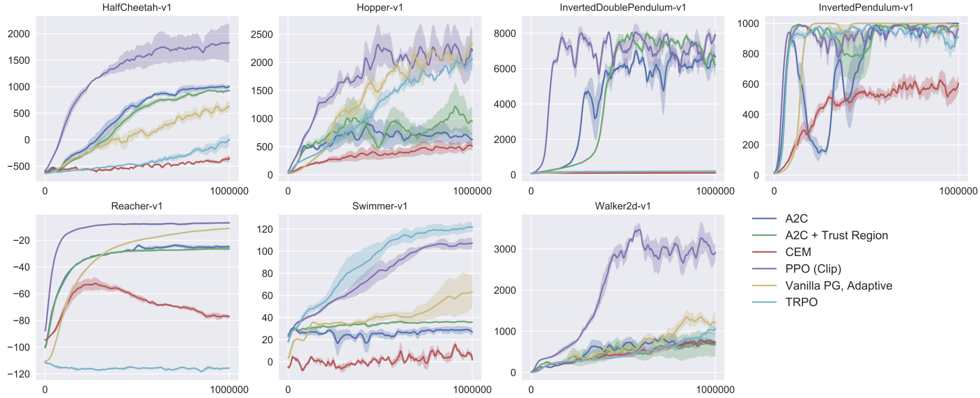


Figure 3: Comparison of several algorithms on several MuJoCo environments, training for one million timesteps.

LLM | RL from Human Feedback

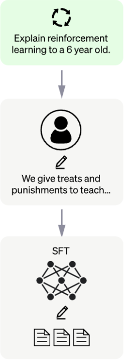
Step 1

Collect demonstration data and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3.5 with supervised learning.



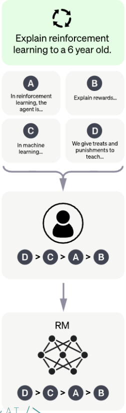
Step 2

Collect comparison data and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

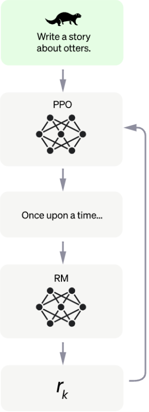
A new prompt is sampled from the dataset.

The PPO model is initialized from the supervised policy.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



Notation & Terminology

- MDP $M = (\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma)$, state $s \in \mathcal{S}$, action $a \in \mathcal{A}$, $r \in \mathcal{R}$ reward $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$, discount factor $\gamma \in [0, 1)$, state-transition prob. $p(s' \mid s, a)$, $p : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$
- Reward function: $r(s, a) = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$ and analogously for a trajectory $\tau \sim (s_0, a_0, s_1, a_1, \ldots, s_t, a_t)$ according to $r(\tau)$
- Policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$: With stochastic policy $\pi(a \mid s)$ for the agent behavior
- Return: $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$ discounted future reward
- Action-value function / Q-function: $Q^\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$ following policy π
- State-value function: $V^\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$ when following policy π
- Advantage function: $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$ state-value baseline subtracted Q-function to reduce variance
- Performance measure: $J(\theta)$ and estimated version $\widehat{J(\theta)}$

Policy-based vs value-based comparison

Policy-based:

- might be favorable for continuous action spaces
- most policy gradient methods are model-free and on-policy
- might converge faster (to local minima)
- gradient step in the direction of maximizing the return and performs direct action selection

Value-based:

- more sample efficient than policy-based methods
- many value-based methods are off-policy methods
- computes the expectations over a value function of states / not used for action selection (action is indirectly retrieved by maximum over Q values)

POLICY GRADIENT DERIVATION

Goal of RL

The goal of reinforcement learning: find a policy π^* such

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\ \mathbb{E}_\pi\left[G_0 \mid s_0\right]$$

And with $\mathbb{E}_\pi\left[G_0 \mid s_0\right]$ we mean:

$$\mathbb{E}_\pi\left[G_0 \mid s_0\right] = \Bigg(p(s_0) \sum_{a \in \mathcal{A}(s_0)} \pi(a \mid s_0) \sum_{r' \in \mathcal{R}(s_0, a)} p(r' \mid s_0, a)\ r'$$
$$+ \sum_t \sum_{s' \in \mathcal{S}} p(s' \mid s, a) \sum_{a' \in \mathcal{A}(s')} \pi(a' \mid s') \sum_{r'' \in \mathcal{R}(s', a')} p(r'' \mid s', a')\ r''\Bigg)$$

Probability of a Trajectory τ

To easy the notation, we define $p_\pi(\tau)$ as the probability of a trajectory $\tau = \{(s_i, a_i)\}_{0 \leqslant i \leqslant T}$ being sampled using policy π:

$$p_\pi(\tau) = p(s_0) \prod_{t=0}^{T} \pi(a_t, s_t) p(s_{t+1} \mid s_t, a_t)$$

Then, the RL goal can be rewritten as

$$\begin{aligned} \pi^* &= \underset{\pi}{\operatorname{argmax}} \; \mathbb{E}_{\tau \sim p_\pi(\tau)} \big[R(\tau) \big] \\ &= \underset{\pi}{\operatorname{argmax}} \; J(\pi) \end{aligned}$$

where $R(\tau) = \sum_t R_{t+1}$ is the random variable of the return for a given trajectory τ, and $J(\pi)$ the performance of a policy π.

Optimizing the Policy I

For a parametrized policy π_θ, the objective is

$$\begin{aligned}\theta^* &= \underset{\theta}{\operatorname{argmax}}\ \mathbb{E}_{\tau \sim p_{\pi_\theta(\tau)}}\big[R(\tau)\big] \\ &= \underset{\theta}{\operatorname{argmax}}\ J(\theta)\end{aligned}$$

We could maximize the expected reward using gradient ascent, and updating the parameters using the following update rule:

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\theta_t)$$

Optimizing the Policy II

Computing the gradient $\nabla_\theta J(\theta_t)$ is tricky because it depends on both the action selection (directly determined by π_θ) and the stationary distribution of states following the target selection behavior (indirectly determined by π_θ)

Policy Gradient I

- Learning a policy π_θ directly based on some scalar performance measure $J(\theta)$ [Sutton and Barto, 1998]:

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta \widehat{J(\theta)}$$

- The derivation of Policy Gradient for on-policy

$$\nabla_\theta \mathbb{E}_{\pi_\theta}[r(\tau)] = \mathbb{E}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)]$$

Policy Gradient II

- **Score Function**: The log derivative trick (sometimes likelihood ratio or score ratio) is [Shakir, 2015]:

$$\nabla_\theta \log p(\boldsymbol{x};\theta) = \frac{1}{p(\boldsymbol{x};\theta)} \nabla_\theta p(\boldsymbol{x};\theta) = \frac{\nabla_\theta p(\boldsymbol{x};\theta)}{p(\boldsymbol{x};\theta)}$$

- **Score function estimator** for the gradient

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{p(\boldsymbol{x};\theta)}[f(\boldsymbol{x})] &= \nabla_\theta \int p(\boldsymbol{x};\theta) f(\boldsymbol{x}) dx \\
&= \int \frac{p(\boldsymbol{x};\theta)}{p(\boldsymbol{x};\theta)} \nabla_\theta p(\boldsymbol{x};\theta) f(\boldsymbol{x}) dx \\
&= \int p(\boldsymbol{x};\theta) \nabla_\theta \log p(\boldsymbol{x};\theta) f(\boldsymbol{x}) dx \\
&= \mathbb{E}_{p(\boldsymbol{x};\theta)}[\nabla_\theta \log p(\boldsymbol{x};\theta) f(\boldsymbol{x})]
\end{aligned}$$

Policy Gradient III

- Working with samples and finite discrete timesteps:

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{a_t \sim \pi_\theta, s_t \sim p(s)} \left[\left(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right) \left(\sum_{t=0}^{T} r(s_t, a_t) \right) \right] \\
&\approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_{t,i} \mid s_{t,i}) \right) \left(\sum_{t=0}^{T} r(s_{t,i}, a_{t,i}) \right)
\end{aligned}
$$

Intuition

- The term $\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t})$ is the maximum log likelihood.
- It measures the likelihood of the current trajectory given the policy π_θ.
- The term $r(\tau) = \sum_{t=0}^{T} r_{i,t+1}$ is the return for a given trajectory τ.
- The gradient $\nabla_\theta \mathbb{E}_{\pi_\theta}\left[R(\tau)\right]$ increases the likelihood of high return trajectories, and decreases likelihood of trajectories with negative returns.

REINFORCE

Algorithm REINFORCE (on-policy)

Initialize policy $\pi_\theta(\cdot)$ with random weights, define step size α

repeat:

Generate episode $\{s_0, a_0, R_1, s_1 \ldots, s_{T-1}, a_{T-1}, R_T, s_T\} \sim \pi_\theta(\cdot)$

$\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(a_{t,i} \mid s_{t,i})\right) \left(\sum_t r(s_{t,i}, a_{t,i})\right)$

$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Intuition:

- good actions are made more likely
- bad actions are made less likely
- “trial and error” approach

REINFORCE Issues

- High variance / noisy gradients $\rightarrow$ unstable learning
- Slow convergence
- Sample inefficient
- Easily collapses to bad solutions
- Problems with exploration

Reducing Variance I - Causality Trick

- **First**, policies at time t' cannot affect reward at time t when $t < t'$, allowing us to use the "reward-to-go" [Levine, 2020]:

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\tau \sim (p(s), \pi_\theta(\cdot|s))} \left[\nabla_\theta \log \pi_\theta(\tau) r(\tau) \right] \\
&\approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_{t,i} \mid s_{t,i}) \left(\sum_{t=0}^{T} r(s_{t,i}, a_{t,i}) \right) \\
&\approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_{t,i} \mid s_{t,i}) \left(\sum_{t'=t}^{T} r(s_{t',i}, a_{t',i}) \right) \\
&\approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_{t,i} \mid s_{t,i}) \hat{Q}^\pi(s_{t,i}, a_{t,i})
\end{aligned}
$$

Reducing Variance II - Baselines

- **Second**, we can subtract any constant b from our reward function that is independent of our actions:

$$\nabla_\theta J(\theta) \approx \frac{1}{n} \sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau) \left[r(\tau) - b\right]$$

$$\begin{aligned}
\mathbb{E}_{\pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) b] &= \int \pi_\theta(\tau) \frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} b \, d\tau \\
&= \int \nabla_\theta \pi_\theta(\tau) b \, d\tau \\
&= b \nabla_\theta \int \pi_\theta(\tau) \, d\tau = b \nabla_\theta 1 = 0
\end{aligned}$$

- Meaning, we can use the advantage function $A(s,a) = Q(s,a) - V(s)$ instead of the Q-function by subtracting the value function as a baseline.

Reducing Variance III - Analysis

$$\mathrm{Var}[x] = \mathbb{E}[x^2] - \mathbb{E}[x]^2$$

$$\begin{aligned}
\mathrm{Var}\big[\nabla_\theta \log p_{\pi_\theta}(\tau)\big(R(\tau) - {\color{blue}b}\big)\big] &= \mathbb{E}_{\tau\sim p_{\pi_\theta}}\left[\left(\nabla_\theta \log p_{\pi_\theta}(\tau)\big(R(\tau) - {\color{blue}b}\big)\right)^2\right] \\
&\quad - \underbrace{\mathbb{E}_{\tau\sim p_{\pi_\theta}}\left[\left(\nabla_\theta \log p_{\pi_\theta}(\tau)\big(R(\tau) - {\color{blue}b}\big)\right)\right]}_{\text{Baselines are unbiased}}^2 \\
&= \mathbb{E}_{\tau\sim p_{\pi_\theta}}\left[\left(\nabla_\theta \log p_{\pi_\theta}(\tau)\big(R(\tau) - {\color{blue}b}\big)\right)^2\right] \\
&\quad - \mathbb{E}_{\tau\sim p_{\pi_\theta}}\left[\left(\nabla_\theta \log p_{\pi_\theta}(\tau)\big(R(\tau)\big)\right)\right]^2
\end{aligned}$$

Actor-Critic Methods I

- **Recall**: We want to increase "good" action and decrease "bad" actions and reduce variance by subtracting a baseline.
- The baseline can have various values as long as it has zero expectation or independent of the actions

$$\begin{aligned}
\nabla J(\theta) &= \mathbb{E}_{\pi_\theta}\left[\nabla \log \pi_\theta(a \mid s) G_t\right] && \text{REINFORCE} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla \log \pi_\theta(a \mid s) Q^\pi(s,a)\right] && \text{Q Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla \log \pi_\theta(a \mid s) A^\pi(s,a)\right] && \text{Advantage Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla \log \pi_\theta(a \mid s) \delta(\cdot)\right] && \text{TD Actor-Critic}
\end{aligned}$$

[Fragkiadaki and Mitchell, 2018], [Weng, 2018]

Actor-Critic Methods II

- The “Critic” estimates either the action-value $Q(s, a)$ or state-value $V(s)$ function.
- The “Actor” updates the policy distribution in the direction suggested by the Critic.
- We prefer to use the advantage function for the Critic, because it offers nice intuitive properties and smaller / more stable gradients:

$$
\begin{aligned}
Q(s_t, a_t) &= \mathbb{E}\left[R_{t+1} + \gamma V(s_{t+1})\right] \\
A(s_t, a_t) &= Q(s_t, a_t) - V(s_t) \\
&= R_{t+1} + \gamma V(s_{t+1}) - V(s_t)
\end{aligned}
$$

only requiring one estimator for the value function $V(s)$.

Policy Gradient Derivation with Importance Sampling

What does happen when trajectory samples comes from a different policy (or from an old policy stored in a buffer)?

We want $\nabla_\theta \mathbb{E}_{\tau \sim p_{\pi_\theta}(\tau)}\big[R(\tau)\big]$, but we sample with μ instead of π_θ.

Importance sampling:

$$
\begin{aligned}
\mathbb{E}_{x \sim p(x)}[f(x)] &= \int p(x) f(x)\,\mathrm{d}x \\
&= \int \frac{q(x)}{q(x)} p(x) f(x)\,\mathrm{d}x \\
&= \int q(x) \frac{p(x)}{q(x)} f(x)\,\mathrm{d}x \\
&= \mathbb{E}_{x \sim q(x)}\Big[\frac{p(x)}{q(x)} f(x)\Big]
\end{aligned}
$$

<extensity^AI/>

Policy Gradient Derivation with Importance Sampling

$$J(\theta) = \mathbb{E}_{\tau \sim p_\mu(\tau)}\left[\frac{p_{\pi_\theta}(\tau)}{p_\mu(\tau)} R(\tau)\right]$$

$$\begin{aligned}\frac{p_{\pi_\theta}(\tau)}{p_\mu(\tau)} &= \frac{p(s_0)\prod_{t=0}^{T}\pi_\theta(a_t \mid s_t)p(s_{t+1} \mid s_t, a_t)}{p(s_0)\prod_{t=0}^{T}\mu(a_t \mid s_t)p(s_{t+1} \mid s_t, a_t)} \\ &= \frac{\prod_{t=0}^{T}\pi_\theta(a_t \mid s_t)}{\prod_{t=0}^{T}\mu(a_t \mid s_t)}\end{aligned}$$

Importance Sampling Recap

- **Recall**: on-policy learning has poor sample efficiency / old collected samples are not reusable after an update.
- When sampling episodes and updating one needs to recollect samples to perform the next update.
- Importance sampling can be used to estimate the value functions for a policy π with samples collected previously from an older policy $\bar{\pi}$.
- If both policies are close enough, this allows us to reuse the old samples to recalculate the total rewards.

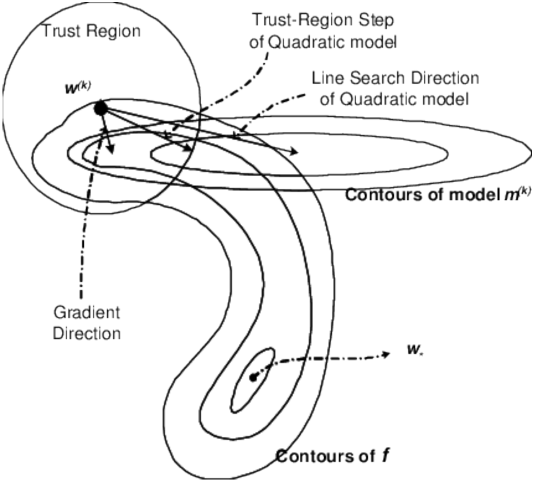
Importance Sampling Solution

- With importance sampling, one can correct and re-weight samples by including the probability ratio between old and new policy:

$$\mathbb{E}_{a_t \sim \bar{\pi}_{\theta_{\text{old}}}(\cdot), s_t \sim p(s)} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \left(\prod_{t'=0}^{t} \frac{\pi_\theta(a_{t'} \mid s_{t'})}{\bar{\pi}_{\theta_{\text{old}}}(a_{t'} \mid s_{t'})} \right) \left(\sum_{t'=t}^{T} r(s_{t'}, a_{t'}) \right) \right]$$

- The generated samples come from the old policy, while we use our new policy to compute the updates.
- We can now collect multiple trajectories with a policy and perform multiple updates from a buffer.

Trust Region Methods I - TRPO



Trust Region Methods II - TRPO

- Furthermore, we can define a new objective with a constraint to ensure that the new policy remains ϵ close to our old policy [Schulman et al., 2015]:

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\bar{\pi}_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t^\pi \right] \\ \text{subject to} \quad & \hat{\mathbb{E}}_t \left[\mathrm{KL}[\bar{\pi}_{\theta_{\mathsf{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)] \right] \leqslant \epsilon \end{aligned}$$

- where $\mathrm{KL}[\cdot]$ is the Kullback–Leibler divergence between old and new policy and $\hat{A}_t$ an estimator of the advantage function at timestep t.
- This **trust region** helps us not to take over-optimistic actions that hurt the training progress.

Trust Region Methods III - TRPO

- The constrained optimization problem can be approximately solved using the conjugate gradient algorithm (numerical solution for a system of linear equations), after making a linear approximation to the objective and a quadratic approximation to the constraint.
- Instead of solving a constrained optimization problem, TRPO proposes a surrogate objective using a penalty term:

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\bar{\pi}_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t^\pi - \beta \text{KL}[\bar{\pi}_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)] \right]$$

with β being a regularization constant.

Trust Region Methods III - PPO

- Since it is difficult to find a proper β for various problems and computing $\mathrm{KL}[\cdot]$ is expensive, PPO suggests a “Clipped Surrogate Objective” [Schulman et al., 2018]:

$$\hat{\mathbb{E}}_t \left[\min\left(\varphi_t(\theta)\hat{A}_t, \mathrm{clip}(\varphi_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t^{\pi}\right)\right]$$

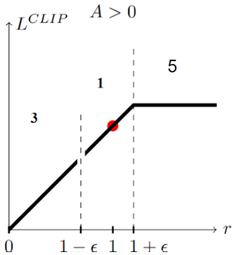
$$\varphi_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\bar{\pi}_{\theta_{\mathrm{old}}}(a_t \mid s_t)}$$

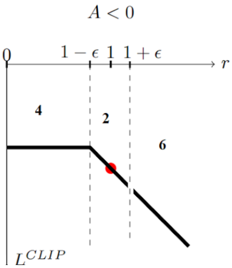
  with ϵ as a hyperparameter.

- The first term is the same as the TRPO surrogate objective without the constraint and the second term creates a lower bound (i.e., pessimistic bound) on the unclipped objective.

Trust Region Methods IV - PPO

	$p_t(\theta) \quad > 0$	A_t	Return Value of *min*	Objective is Clipped	Sign of Objective	Gradient
1	$p_t(\theta) \in [1-\epsilon, 1+\epsilon]$	$+$	$p_t(\theta)A_t$	no	$+$	✓
2	$p_t(\theta) \in [1-\epsilon, 1+\epsilon]$	$-$	$p_t(\theta)A_t$	no	$-$	✓
3	$p_t(\theta) < 1-\epsilon$	$+$	$p_t(\theta)A_t$	no	$+$	✓
4	$p_t(\theta) < 1-\epsilon$	$-$	$(1-\epsilon)A_t$	yes	$-$	**0**
5	$p_t(\theta) > 1+\epsilon$	$+$	$(1+\epsilon)A_t$	yes	$+$	**0**
6	$p_t(\theta) > 1+\epsilon$	$-$	$p_t(\theta)A_t$	no	$-$	✓





Maximum Entropy

- **Recall**: We need to encourage exploration and help prevent early convergence to sub-optimal policies.
- We can augment the standard maximum RL objective with an entropy regularization term:

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)(A^\pi(s_t, a_t) + \beta \nabla_\theta \mathcal{H}(\pi_\theta(\cdot \mid s_t))\right]$$

- The resulting policies can serve as a good initialization for fine-tuning to a more specific behavior.
- Provides a better exploration mechanism for seeking out the best mode in a multimodal reward landscape.
- Resulting policies are more robust in the face of adversarial perturbations [Shi et al., 2019].

Trust Region Methods IV - PPO

Algorithm PPO, Actor-Critic

Initialize actor $\pi_\theta(\cdot)$ and critic $\nu_\omega(\cdot)$ with random weights and create an empty buffer $\mathcal{D}$, define step sizes $\alpha_{1..2}$

for $t = 0$ **to** T **do**:

- Generate and store n transition tuples from timestep t
- $\mathcal{D} \leftarrow \mathcal{D} \cup \left\{s_i, a_i, R_{i+1}, s_{i+1}, \log \bar{\pi}_{\theta_{\text{old}}}(\cdot \mid s_i) \mid \bar{\pi}_{\theta_{\text{old}}}\right\}_{i=t}^{t+n}$
- **for** k update steps **do**:
  - Sample a batch $\mathcal{B}$ of transitions from the buffer $\mathcal{D}$
  - $\left(s_j, a_j, R_{j+1}, s_{j+1}, \bar{\pi}_{\theta_{\text{old}}}(\cdot \mid s_j)\right) \sim \mathcal{B}_\mathcal{D}$
  - Compute target function $v_{\text{target}} = R_{j+1} + \gamma \nu_\omega(s_{j+1})$
  - Compute advantage function $\hat{A}_j = v_{\text{target}} - \nu_\omega(s_j)$
  - Compute importance sampling ratio $\widehat{\varphi}_j(\theta) = \log \frac{\pi_\theta(a_j|s_j)}{\bar{\pi}_{\theta_{\text{old}}}(a_j|s_j)}$
  - $\mathcal{L}^{\text{CLIP}}(\theta, \omega) = -\hat{\mathbb{E}}_j \left[\min\left(\widehat{\varphi}_j(\theta)\hat{A}_j(\omega), \text{clip}(\widehat{\varphi}_j(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_j(\omega)\right)\right]$
  - $\mathcal{L}^{\text{VAL}}(\omega) = \left\|v_{\text{target}} - \nu_\omega(s_j)\right\|^2$, $\mathcal{L}^{\text{ENT}}(\theta) = -\mathcal{H}(\pi_\theta(\cdot \mid s_j))$
  - $\mathcal{L}(\theta, \omega) = \mathcal{L}^{\text{VAL}}(\omega) + \mathcal{L}^{\text{CLIP}}(\theta, \omega) + \mathcal{L}^{\text{ENT}}(\theta)$
  - $\theta \leftarrow \theta - \alpha_1 \nabla_\theta \mathcal{L}(\theta, \omega)$, $\omega \leftarrow \omega - \alpha_2 \nabla_\omega \mathcal{L}(\theta, \omega)$
- reset buffer $\mathcal{D}$

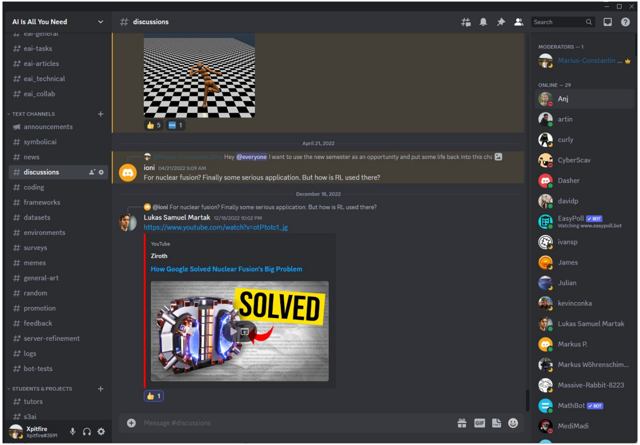
Recommendations

- Stay in shape (physically, as well as your tensors - the former keeps you healthy, the latter may save you a few hours of debugging)
- Handle the high variance in your reward (works poorly for large value ranges, e.g. values $> +/-10$)
- Optionally impl. GAE [Schulman et al., 2016]
- Try shallow network architectures
- Play around with hyperparameters after you have something working
- Change only one thing at a time and try out an experiment
- The critic has to learn faster than the actor

FAQ - Proximal Policy Optimization (PPO)

- **Keep it simple:** Shallow networks often work better than deep ones.
- **Hyperparameters:** Focus on tuning learning rate, batch size, and clipping epsilon.
- **Reward scaling:** Normalize rewards; PPO struggles with large value ranges ($> +/-10$).
- **Monitor KL divergence:** Ensure policy updates aren't too large.
- **Implement GAE:** Use Generalized Advantage Estimation for better performance.
- **Troubleshooting:** If the agent isn't improving:
  - Review reward function and network architecture
  - Analyze training logs for instabilities
  - Collaborate with peers or consult online communities
- **Time management:** Allow sufficient time for experimentation and debugging.

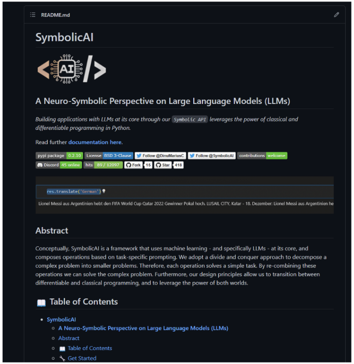
AI Is All You Need Discord Server

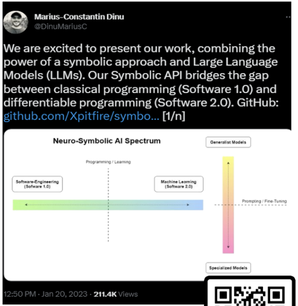


Invite Link: https://discord.gg/azDQxCHeDA

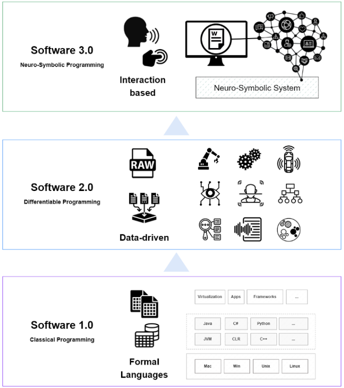
<extensity AI/>

ExtensityAI





SCAN ME



www.extensity.ai

APPENDIX K

Generalized Advantage Estimation (GAE)

GAE [Schulman et al., 2016] uses the discounted λ-advantage function to weight and drive the gradients.

- The advantage function $A(s_t, a_t)$ yields almost the lowest possible variance.
- Intuition: A step in the policy gradient direction should increase the probability of better-than-average actions and decrease the probability of worse-than average actions.
- The discounting factor γ reduce the variance by downweighting rewards corresponding to delayed effects, at the cost of introducing bias.
- The generalized advantage estimator GAE(γ, λ) is defined as the exponentially-weighted average of k-step advantage estimators (similar to TD(λ).

GAE I

Advantage k-steps

$$
\begin{aligned}
\hat{A}_t^{(1)} &:= \delta_t^V && = -V(s_t) + r_t + \gamma V(s_{t+1}) \\
\hat{A}_t^{(2)} &:= \delta_t^V + \gamma \delta_{t+1}^V && = -V(s_t) + r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2}) \\
\hat{A}_t^{(3)} &:= \delta_t^V + \gamma \delta_{t+1}^V + \gamma^2 \delta_{t+2}^V && = -V(s_t) + r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \\
& && + \gamma^3 V(s_{t+3})
\end{aligned}
$$

$$
\begin{aligned}
\hat{A}_t^{(k)} := \sum_{l=0}^{k-1} \gamma^l \delta_{t+l}^V = -V(s_t) + r_t + \gamma r_{t+1} + \dots \\
+ \gamma^{k-1} r_{t+k-1} + \gamma^k V(s_{t+k})
\end{aligned}
$$

GAE II

The generalized advantage estimator $\mathrm{GAE}(\gamma, \lambda)$ is defined as the exponentially-weighted average of these k-step estimators:

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

GAE III - Special Cases

There are two notable special cases of this formula, obtained by setting $\lambda = 0$ and $\lambda = 1$.

$$\begin{aligned}
\mathrm{GAE}(\gamma, 0) : \quad \hat{A}_t &:= \delta_t && = r_t + \gamma V(s_{t+1}) - V(s_t) \\
\mathrm{GAE}(\gamma, 1) : \quad \hat{A}_t &:= \sum_{l=0}^{\infty} \gamma^l \delta_{t+l} && = \sum_{l=0}^{\infty} \gamma^l r_{t+l} - V(s_t)
\end{aligned}$$

REFERENCES

References I

[Fragkiadaki and Mitchell, 2018] Fragkiadaki, K. and Mitchell, T. (2018).
Deep reinforcement learning and control.

[Levine, 2020] Levine, S. (2020).
Policy gradients CS 285 | Deep Reinforcement Learning.

[Schulman et al., 2015] Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. (2015).
Trust region policy optimization.
In *32st International Conference on Machine Learning (ICML)*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897. PMLR.

References II

[Schulman et al., 2016] Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2016).
High-dimensional continuous control using generalized advantage estimation.
In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[Schulman et al., 2018] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2018).
Proximal policy optimization algorithms.
ArXiv, 1707.06347.

References III

[Shakir, 2015] Shakir, M. (2015).
Log derivative trick.
Technical report, The Spectator.

[Shi et al., 2019] Shi, W., Song, S., and Wu, C. (2019).
Soft policy gradient method for maximum entropy deep reinforcement learning.

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998).
Reinforcement Learning - An Introduction.
MIT Press, Cambridge, MA.

[Weng, 2018] Weng, L. (2018).
Policy gradient algorithms.
Technical report, Lil'Log.