A2C and from TRPO to PPO

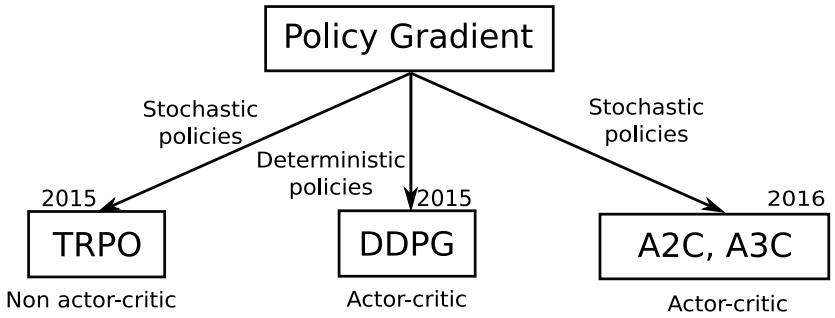
Olivier Sigaud

Sorbonne Université
http://people.isir.upmc.fr/sigaud

Outline

- Actor-critic approach: A3C, A2C
- Then, more PG with baselines: TRPO and ACKTR
- Three aspects distinguish TRPO:
  - Surrogate return objective
  - Natural policy gradient
  - Conjugate gradient approach
- Differences in ACKTR:
  - Approximate second order gradient descent (Hessian)
  - Using Kronecker Factored Approximated Curvature
- Then PPO (a quick overview of two versions)

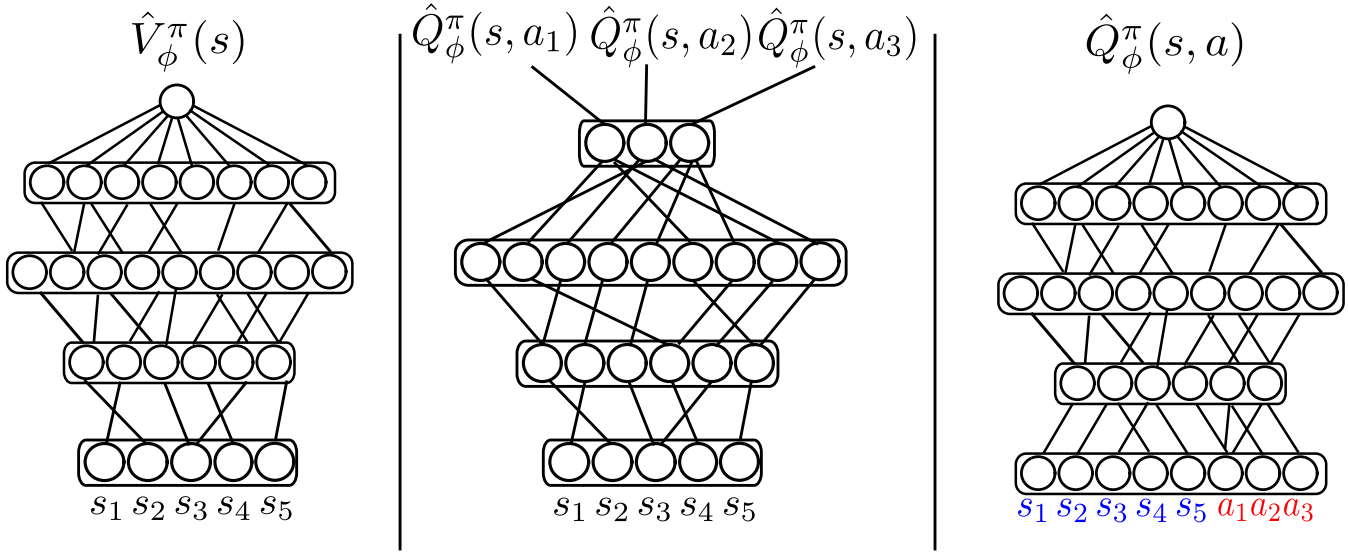
Advantage Actor Critic (A2C)



- A crucial move from Policy Gradient methods to Actor-Critic methods
- The earliest actor-critic algorithm of the deep RL era using stochastic policies
- It directly derives from the basic Policy Gradient method
- The critic is learned using bootstrap, which makes it an actor-critic algorithm
- The A2C paper focuses more on A3C, an asynchronous version where several agents generate data without using a replay buffer
- A2C can be seen as a simplified version of A3C with a single agent

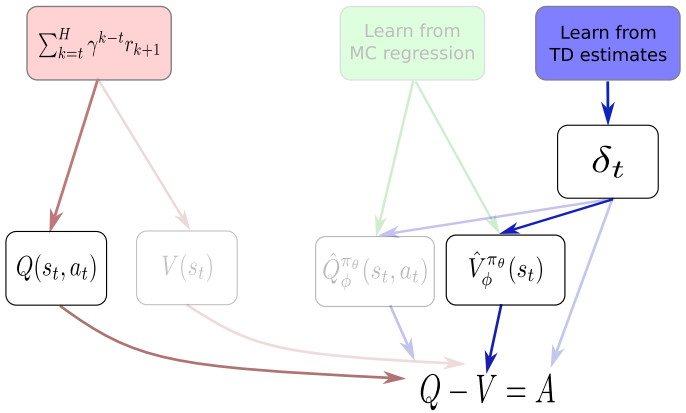
Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. (2016) Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783*

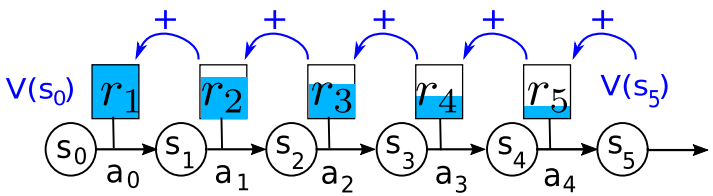
Choice of a V critic



- Main point: By contrast with $Q(s,a)$, $V(s)$ can be estimated in the same way irrespective of using discrete or continuous actions
- $\hat{V}^{\pi}_{\boldsymbol{\phi}}$ is smaller, but not necessarily easier to estimate (implicit max over actions)
- Temporal difference error: $\delta = [r(\mathbf{s}_t) + \gamma V^i_{\boldsymbol{\phi}}(\mathbf{s}_{t+1}) - V^i_{\boldsymbol{\phi}}(\mathbf{s}_t)]$
- Standard update rule: $V^{i+1}_{\boldsymbol{\phi}}(\mathbf{s}_t) \leftarrow V^i_{\boldsymbol{\phi}}(\mathbf{s}_t) + \alpha\delta$

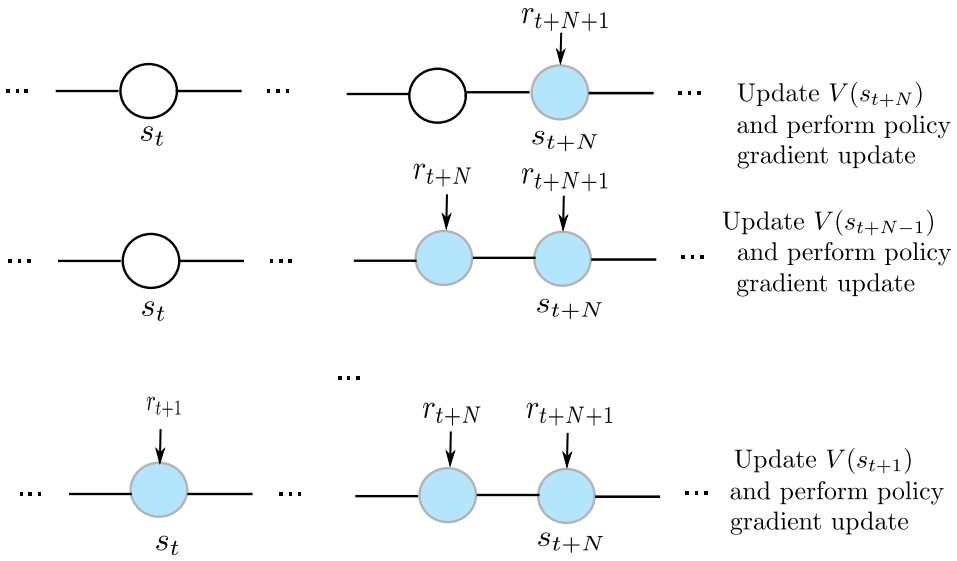
Advantage function calculation





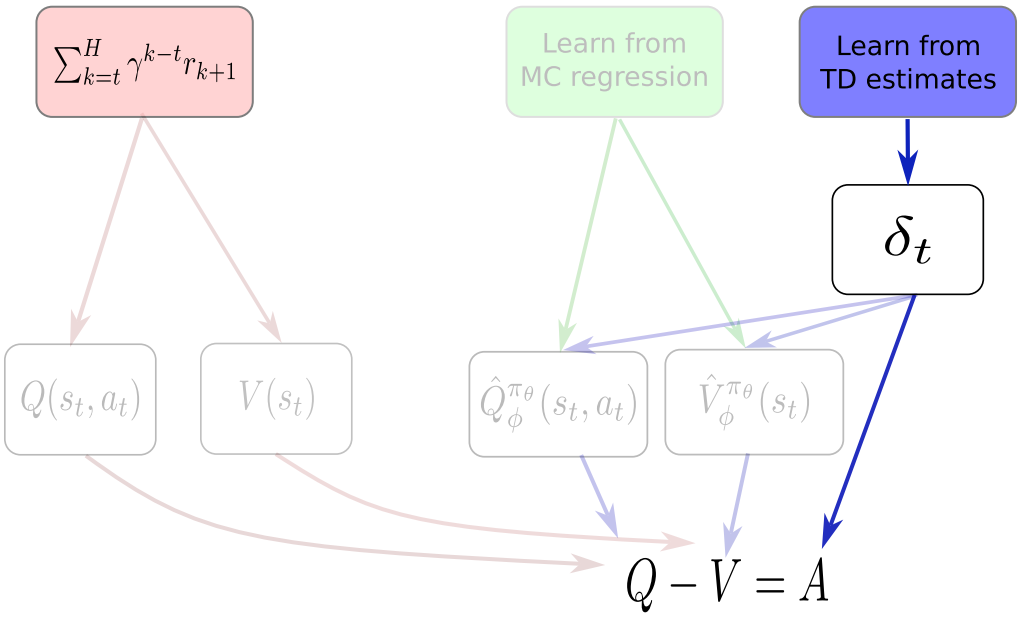
- To perform policy gradient updates, one needs to compute $\hat{A}_{\phi}(\mathbf{s}_t, \mathbf{a}_t)$
- By definition, $A(\mathbf{s}_t, \mathbf{a}_t) = Q(\mathbf{s}_t, \mathbf{a}_t) - V(\mathbf{s}_t)$
- A2C computes the advantage with $\hat{A}_{\phi}(\mathbf{s}_t, \mathbf{a}_t) = R_t(\mathbf{s}_t) - V_{\phi}(\mathbf{s}_t)$
- $R_t(\mathbf{s}_t) = \sum_{i=0}^{N-1} \gamma^i r_{t+i+1} + \gamma^N V_{\phi}(\mathbf{s}_{t+N})$ is the return of the current N-step trajectory from state $\mathbf{s}_t$
- $R_t(\mathbf{s}_t)$ can be seen as an approximate of $Q(\mathbf{s}_t, \mathbf{a}_t)$ computed along one trajectory
- Actually, computed on N steps rather than the full trajectory

N-step updates



- The agent performs N steps in the environment (or less if the episode stops earlier in the episodic case) before each update
- At each update, the agent has collected up to N states and rewards
- It can update the value of the last state using the last reward, the value of the second last step with two rewards
- And so on up to the first state of the current collection
- It updates both the critic and the policy at each update
- Straightforward in BBRL

Alternative implementation



- With this approach, the advantage can be computed out of current step information only
- Makes it possible to add a replay buffer and make it more off-policy
- In *bbrl_algos*, use the GAE version

Policy Gradient updates

- The standard Policy Gradient update is:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t \sim \pi_{\boldsymbol{\theta}}(.)}[\nabla_{\boldsymbol{\theta}}[\log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t^{(i)} | \mathbf{s}_t^{(i)})] \hat{A}_{\boldsymbol{\phi}}(\mathbf{s}_t, \mathbf{a}_t)]$$

- But to favor exploration, A2C adds an entropy term to the gradient calculation
- Thus the policy update rule is:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t \sim \pi_{\boldsymbol{\theta}}(.)}[\nabla_{\boldsymbol{\theta}}[\log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t^{(i)} | \mathbf{s}_t^{(i)})(R_t - V_{\boldsymbol{\phi}}(\mathbf{s}_t)) - \beta \mathcal{H}(\pi_{\boldsymbol{\theta}}(\mathbf{s}_t))]]$$

- where $\mathcal{H}(\pi_{\boldsymbol{\theta}}(\mathbf{s}_t))$ is the entropy of policy $\pi_{\boldsymbol{\theta}}$ at state $\mathbf{s}_t$.
- Note that A2C adds entropy in the update of the actor, but outside the critic, whereas SAC adds it in the critic target, which has a deeper impact.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A. Abbeel, P. et al. (2018) Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*

Summary: main distinguishing features

- To perform policy gradient, you need the advantage function
- Computes the advantage function as value function minus the return of the current N-step trajectory
- Adds entropy regularization to favor exploration in the gradient calculation step
- Uses N-step updates
- Does not use a replay buffer
- Note that A2C is Actor-Critic, but on-policy, so one cannot equate Actor-Critic and off-policy

TRPO

Surrogate return objective

- The standard policy gradient algorithm for stochastic policies is:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_t[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t) \hat{A}^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\phi}}]$$

- This gradient is obtained from differentiating
$Loss^{PG}(\boldsymbol{\theta}) = \mathbb{E}_t[\log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t) \hat{A}^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\phi}}]$
- But we obtain the same gradient from differentiating

$$Loss^{IS}(\boldsymbol{\theta}) = \mathbb{E}_t[\frac{\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_{\boldsymbol{\theta}old}(\mathbf{a}_t|\mathbf{s}_t)} \hat{A}^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\phi}}]$$

  where $\pi_{\boldsymbol{\theta}old}$ is the policy at the previous iteration
- Because $\nabla_{\boldsymbol{\theta}} \log f(\boldsymbol{\theta})|_{\boldsymbol{\theta}old} = \frac{\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})|_{\boldsymbol{\theta}old}}{f(\boldsymbol{\theta}old)} = \nabla_{\boldsymbol{\theta}}(\frac{f(\boldsymbol{\theta})}{f(\boldsymbol{\theta}old)})|_{\boldsymbol{\theta}old}$

- Another view based on importance sampling
- See John Schulmann's Deep RL bootcamp lecture #5
https://www.youtube.com/watch?v=xvRrgxcpaHY (8')

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., & Abbeel, P. (2015) Trust Region Policy Optimization. *CoRR, abs/1502.05477*

The policy gradient is on-policy

- The policy gradient calculation assumes that the training trajectories are obtained from the policy we are optimizing:
- Reminder: we want to find $\text{argmax}_{\boldsymbol{\theta}} \sum_{\tau} P(\tau, \boldsymbol{\theta}) \psi(\tau)$
- We use

$$P(\tau^{(i)}, \boldsymbol{\theta}_{samp}) = \prod_{t=1}^{H} p(s_{t+1}^{(i)} | s_t^{(i)}, a_t^{(i)}) . \pi_{\boldsymbol{\theta}_{samp}}(a_t^{(i)} | s_t^{(i)})$$

- Here, by definition, $\pi_{\boldsymbol{\theta}_{samp}}(a_t^{(i)} | s_t^{(i)})$ is the policy which generated the trajectories
- Then we take the gradient and get the policy gradient formula
- If we want to optimize another policy $\pi_{\boldsymbol{\theta}_{other}}(a_t^{(i)} | s_t^{(i)})$, the derivation is wrong

Importance sampling

$$\begin{aligned}\mathbb{E}_{x\sim\boldsymbol{\theta}_1}[f(x)] &= P(x|\boldsymbol{\theta}_1)f(x) \\ &= \frac{P(x|\boldsymbol{\theta}_1)}{P(x|\boldsymbol{\theta}_2)}P(x|\boldsymbol{\theta}_2)f(x) \\ &= \frac{P(x|\boldsymbol{\theta}_1)}{P(x|\boldsymbol{\theta}_2)}\mathbb{E}_{x\sim\boldsymbol{\theta}_2}[f(x)]\end{aligned}$$

- $\frac{P(x|\boldsymbol{\theta}_1)}{P(x|\boldsymbol{\theta}_2)}$ is the importance sampling term

Importance sampling: application to TRPO

- We sampled data from $\pi_{\boldsymbol{\theta}_{samp}}$
- We want to optimize another policy $\pi_{\boldsymbol{\theta}_{other}}$,
- We can rewrite

$$P(\tau^{(i)}, \boldsymbol{\theta}_{other}) = \prod_{t=1}^{H} p(s_{t+1}^{(i)}|s_t^{(i)}, a_t^{(i)}).\pi_{\boldsymbol{\theta}_{other}}(a_t^{(i)}|s_t^{(i)}).\frac{\pi_{\boldsymbol{\theta}_{samp}}(a_t^{(i)}|s_t^{(i)})}{\pi_{\boldsymbol{\theta}_{samp}}(a_t^{(i)}|s_t^{(i)})}$$

- Or

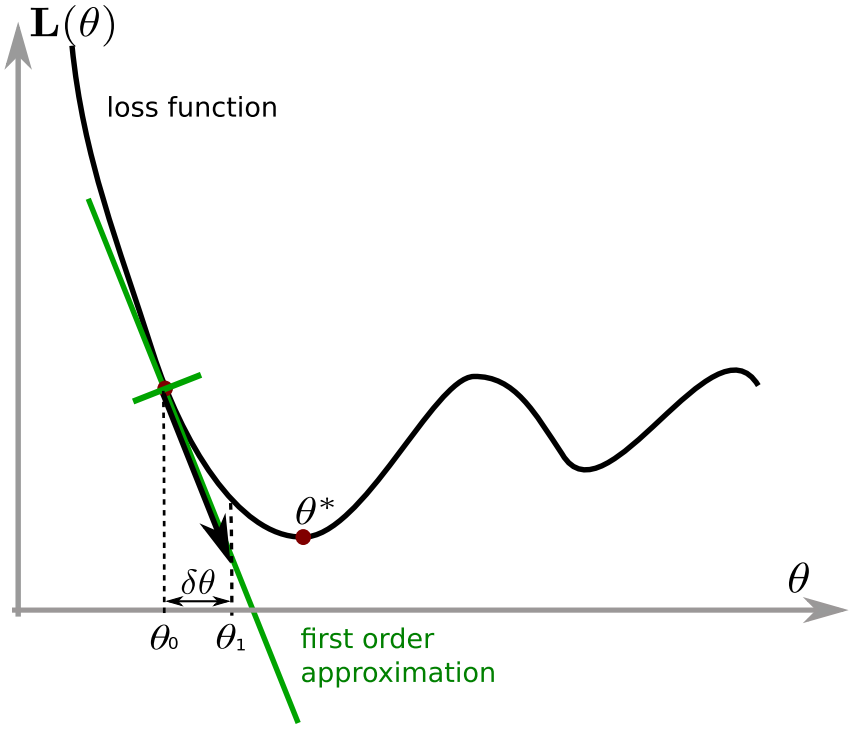
$$P(\tau^{(i)}, \boldsymbol{\theta}_{other}) = \prod_{t=1}^{H} p(s_{t+1}^{(i)}|s_t^{(i)}, a_t^{(i)}).\frac{\pi_{\boldsymbol{\theta}_{other}}(a_t^{(i)}|s_t^{(i)})}{\pi_{\boldsymbol{\theta}_{samp}}(a_t^{(i)}|s_t^{(i)})}.\pi_{\boldsymbol{\theta}_{samp}}(a_t^{(i)}|s_t^{(i)})$$

- And we can get

$$\nabla_{\boldsymbol{\theta}_{other}} J(\boldsymbol{\theta}_{other}) = \frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{H}\frac{\pi_{\boldsymbol{\theta}_{other}}(a_t^{(i)}|s_t^{(i)})}{\pi_{\boldsymbol{\theta}_{samp}}(a_t^{(i)}|s_t^{(i)})}\nabla_{\boldsymbol{\theta}_{samp}} \log \pi_{\boldsymbol{\theta}_{samp}}(a_t^{(i)}|s_t^{(i)})\psi(\tau^{(i)})$$

- The term $\frac{\pi_{\boldsymbol{\theta}_{other}}(a_t^{(i)}|s_t^{(i)})}{\pi_{\boldsymbol{\theta}_{samp}}(a_t^{(i)}|s_t^{(i)})}$ is the importance sampling term
- In TRPO, $\pi_{\boldsymbol{\theta}_{sample}} = \pi_{\boldsymbol{\theta}_{old}}$, $\pi_{\boldsymbol{\theta}_{other}} = \pi_{\boldsymbol{\theta}}$

Trust region



- The gradient of a function is only accurate close to the point where it is calculated
- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is only accurate close to the current policy $\pi_{\boldsymbol{\theta}}$
- Thus, when updating, $\pi_{\boldsymbol{\theta}}$ must not move too far away from a "trust region" around $\pi_{\boldsymbol{\theta}_{old}}$

Kakade, S. & Langford, J. (2002) Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, pages 267–274

Trust Region Policy Optimization

- Theory: monotonous improvement towards the optimal policy (Assumptions do not hold in practice)
- To ensure small steps, TRPO uses a natural gradient update instead of standard gradient
- Minimize Kullback-Leibler divergence to previous policy
-

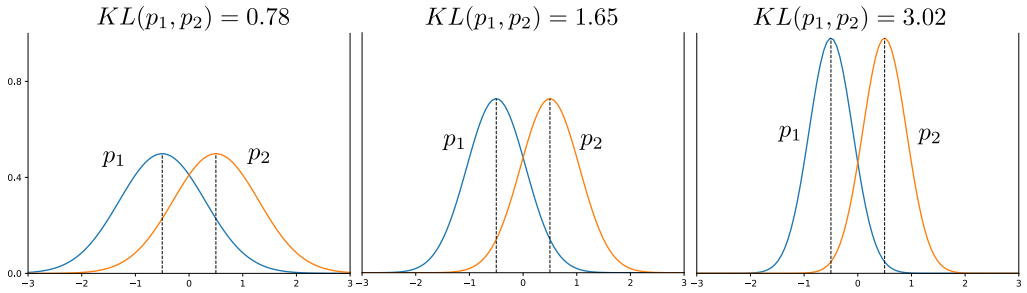
$$\max_{\boldsymbol{\theta}} \mathbb{E}_t\left[\frac{\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_{\boldsymbol{\theta}old}(\mathbf{a}_t|\mathbf{s}_t)} A_{\boldsymbol{\phi}}^{\pi_{\boldsymbol{\theta}old}}(\mathbf{s}_t, \mathbf{a}_t)\right]$$

$$\text{subject to } \mathbb{E}_t[KL(\pi_{\boldsymbol{\theta}old}(.|\mathbf{s})||\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t))] \leq \epsilon$$

- In TRPO, optimization performed using a conjugate gradient method to avoid approximating the Fisher Information matrix

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., & Abbeel, P. (2015) Trust Region Policy Optimization. *CoRR, abs/1502.05477*

Natural Policy Gradient



- One way to constrain two stochastic policies to stay close is constraining their KL divergence
- The KL divergence is smaller when the variance is larger
- Under fixed KL constraint, it is easier to move the mean further away when the variance is large
- Thus the mean policy converges first, then the variance is reduced
- Ensures a large enough amount of exploration noise
- Other properties presented in the Pierrot et al. (2018) paper

Sham M. Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pp. 1531–1538, 2002

Pierrot, T., Perrin, N., & Sigaud, O. (2018) First-order and second-order variants of the gradient descent: a unified framework. *arXiv preprint arXiv:1810.08102*

Advantage estimation

- To get $\hat{A}^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\phi}}$, an empirical estimate of $V^{\pi_{\boldsymbol{\theta}}}(s)$ is needed
- TRPO uses a MC estimate approach through regression, but constrains it (as for the policy):

$$\min_{\boldsymbol{\phi}} \sum_{n=0}^{N} ||V^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\phi}}(s_n) - V^{\pi_{\boldsymbol{\theta}}}(s_n)||^2$$

$$\text{subject to } \frac{1}{N} \sum_{n=0}^{N} \frac{||V^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\phi}}(s_n) - V^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\phi}_{old}}(s_n)||^2}{2\sigma^2} \leq \epsilon$$

- Equivalent to a mean KL divergence constraint between $V^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\phi}}$ and $V^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\phi}_{old}}$

Properties

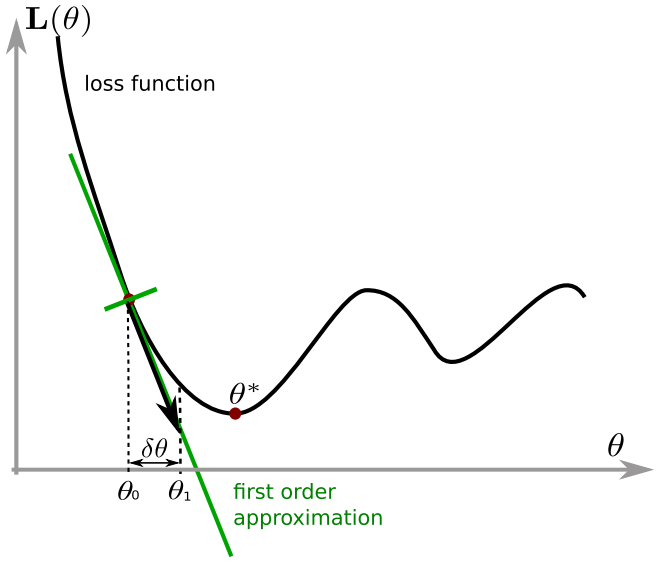
- Moves slowly away from current policy
- Key: use of line search to deal with the gradient step size
- More stable than DDPG, performs well in practice, but less sample efficient
- Conjugate gradient approach not provided in standard tensor gradient librairies, thus not much used
- Greater impact of PPO
- Related work: NAC, REPS

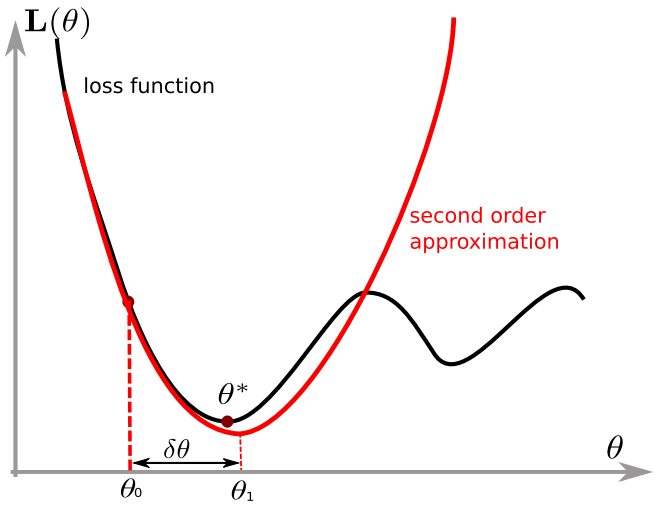

Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71 (7-9):1180–1190, 2008


Jan Peters, Katharina Mülling, and Yasemin Altun. Relative entropy policy search. In *AAAI*, pp. 1607–1612. Atlanta, 2010

ACKTR

First order versus second order derivative





- In first order methods, need to define a step size
- Second order methods provide a more accurate approximation
- They also provide a true minimum, when the Hessian matrix is symmetric positive-definite (SPD)
- In both cases, the derivative is very local
- The trust region constraint applies too

ACKTR

- K-FAC: Kronecker Factored Approximated Curvature: efficient estimate of the gradient
- Using block diagonal estimations of the Hessian matrix, to do better than first order
- ACKTR: TRPO with K-FAC natural gradient calculation
- But closer to actor-critic updates (see PPO)
- The per-update cost of ACKTR is only 10% to 25% higher than SGD
- Improves sample efficiency
- Not much excitement: less robust gradient approximation?
- Next lesson: PPO

Yuhuai Wu, Elman Mansimov, Shun Liao, Roger Grosse, and Jimmy Ba (2017) Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. *arXiv preprint arXiv:1708.05144*

PPO

PPO: Outline

- There are two PPO algorithms
- They are well covered on youtube videos
- So only a quick overview here
- "Easy" implementation (but a lot of tricks), a lot used
- Key question: is it Actor-Critic?

Proximal Policy Optimization (Algorithm 1)

- The conjugate gradient method of TRPO is not available in tensor libraries
- Same idea as TRPO, but uses a soft constraint on trust region rather than a hard one
- Instead of:

$$\max_{\boldsymbol{\theta}} \mathbb{E}_t\left[\frac{\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_{\boldsymbol{\theta}old}(\mathbf{a}_t|\mathbf{s}_t)} A_{\pi_{\boldsymbol{\theta}old}}(\mathbf{s}_t, \mathbf{a}_t)\right]$$

$$\text{subject to } \mathbb{E}_t[KL(\pi_{\boldsymbol{\theta}old}(.|s)||\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t))] \leq \epsilon$$

- Rather use:

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{s\sim\rho,a\sim\pi}\left[\frac{\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_{\boldsymbol{\theta}old}(\mathbf{a}_t|\mathbf{s}_t)} A_{\pi_{\boldsymbol{\theta}old}}(\mathbf{s}_t, \mathbf{a}_t)\right] - \beta\mathbb{E}_{s\sim\rho}[KL(\pi_{\boldsymbol{\theta}old}(.|\mathbf{s})||\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t))]$$

- Makes it possible to use SGD instead of conjugate gradient

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal Policy Optimization Algorithms. arXiv preprint arXiv:1707.06347.

Heess, N., Sriram, S., Lemmon, J., Merel, J., Wayne, G., Tassa, Y., Erez, T., Wang, Z., Eslami, A., Riedmiller, M., et al. (2017). Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*

Proximal Policy Optimization (Algorithm 2)

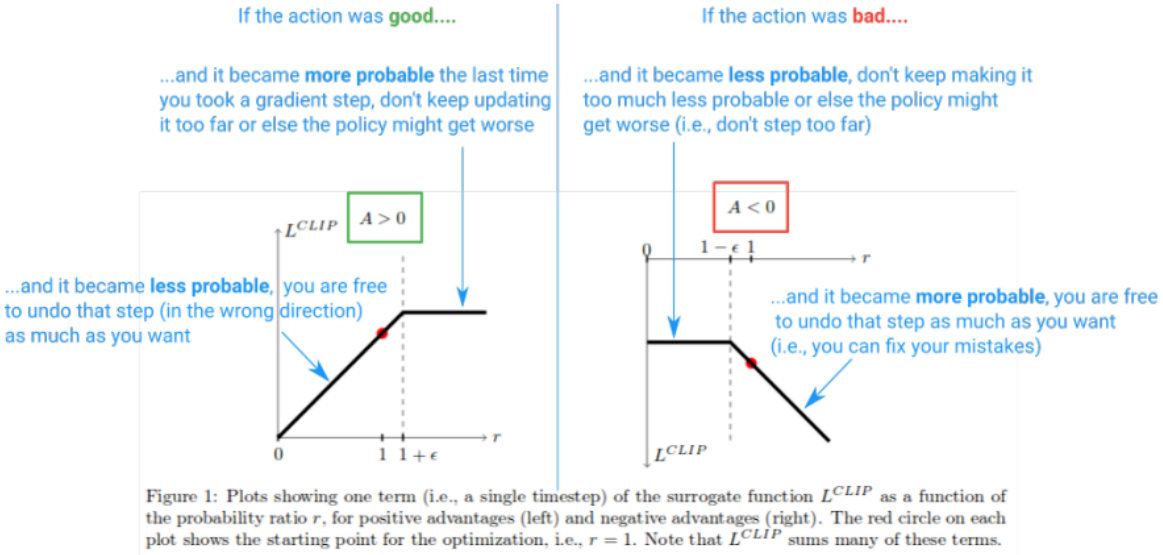


Figure 1: Plots showing one term (i.e., a single timestep) of the surrogate function L^{CLIP} as a function of the probability ratio r, for positive advantages (left) and negative advantages (right). The red circle on each plot shows the starting point for the optimization, i.e., $r = 1$. Note that L^{CLIP} sums many of these terms.

- Image taken from stackoverflow.com
- $\frac{\pi_{\boldsymbol{\theta}}(a|s)}{\pi_{\boldsymbol{\theta}old}(a|s)}$ may get huge if $\pi_{\boldsymbol{\theta}old}$ is very small
- Clipped importance sampling loss (clipping the surrogate objective)

$$r_t(\boldsymbol{\theta}) = \frac{\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_{\boldsymbol{\theta}old}(\mathbf{a}_t|\mathbf{s}_t)}$$

$$L^{CLIP}(\boldsymbol{\theta}) = \mathbb{E}_t[min(r_t(\boldsymbol{\theta})\hat{A}_t, clip(r_t(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$$

- Back-propagate $L^{CLIP}(\boldsymbol{\theta})$ through a policy network

Is PPO actor-critic?

- Improvement over TRPO, thus REINFORCE-like policy update
- But:
  - Algorithm: "PPO, actor-critic style"
  - In the Dota-2 paper: "PPO, a variant of advantage actor-critic, ..."
- What matters is the critic (or baseline) update method
- Uses N-step Generalized Advantage Estimate instead of Monte Carlo
- Thus somewhere between MC and TD (same for ACKTR)
- Other properties:
  - Simpler implementation, better performance than TRPO
  - Does not use a replay buffer $\rightarrow$ more stable, less sample efficient
  - Still on-policy, $\pi_{\boldsymbol{\theta}}$ and $\pi_{\boldsymbol{\theta}old}$ cannot differ much

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019

PPO and A2C

- Like A2C, PPO learns a value function V to compute an advantage
- PPO is very similar to A2C but
- One needs to store the previous policy $\pi_{\boldsymbol{\theta}old}(a|s)$
- The actor loss uses the clipped ratio $\frac{\pi_{\boldsymbol{\theta}}(a|s)}{\pi_{\boldsymbol{\theta}old}(a|s)}$ instead of the log probability $log(\pi_{\boldsymbol{\theta}}(a|s))$
- Several additional context-dependent tricks have been added, see: https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/

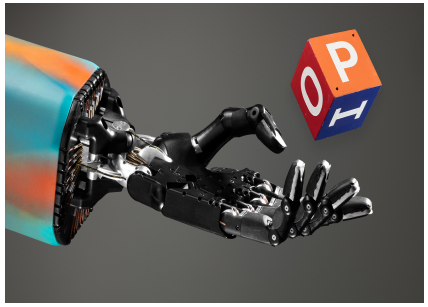
PPO and A2C: In more details

- One can show that A2C is a special case of PPO with specific hyper-parameters
- PPO algorithm (pseudo-code):
- $\pi_{\boldsymbol{\theta}} = \pi_{\boldsymbol{\theta}old}$
- For i in $\{1, \cdots, K\}$
  - $r(\boldsymbol{\theta}) = \frac{\pi_{\boldsymbol{\theta}}}{\pi_{\boldsymbol{\theta}old}}$
  - loss $= \mathbb{E}[r(\boldsymbol{\theta})\hat{A}]$ with clipping or regularization
  - $(\nabla_{\boldsymbol{\theta}}(\text{loss}) = \nabla_{\boldsymbol{\theta}}\mathbb{E}[r(\boldsymbol{\theta})\hat{A}] = \mathbb{E}\frac{\pi_{\boldsymbol{\theta}}}{\pi_{\boldsymbol{\theta}old}}\nabla_{\boldsymbol{\theta}} log\pi_{\boldsymbol{\theta}}\hat{A})$
  - $\pi_{\boldsymbol{\theta}} \leftarrow$ loss
- $\pi_{\boldsymbol{\theta}}$ moves away from $\pi_{\boldsymbol{\theta}old}$ for K iterations
- If $K = 1$, at first iteration $\pi_{\boldsymbol{\theta}} = \pi_{\boldsymbol{\theta}old}$
- We get the A2C loss
- PPO uses GAE. We get A2C if $\lambda = 1$

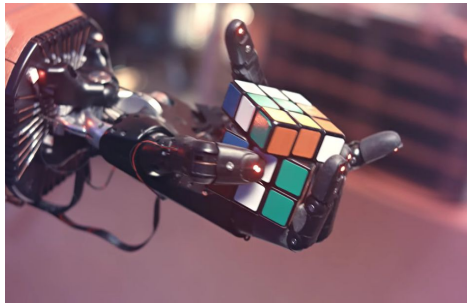
Huang, S., Kanervisto, A., Raffin, A., Wang, W., Ontañón, S., & Dossa, R. F. J. (2022) A2C is a special case of PPO. *arXiv preprint arXiv:2205.09123*

PPO applications

1536 GPU at peak, 10 months for training, 40.000 years



a pool of 384 worker machines, each with 16 CPU cores



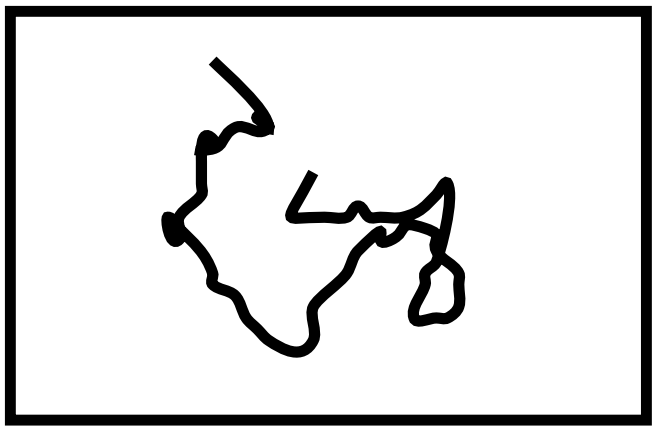
64 V100 GPU + 900 workers, with 32 CPU cores, several months, 13.000 years

- Massive parallel versions of PPO, with dedicated architectures
- Very few teams can afford such engineering and computing effort

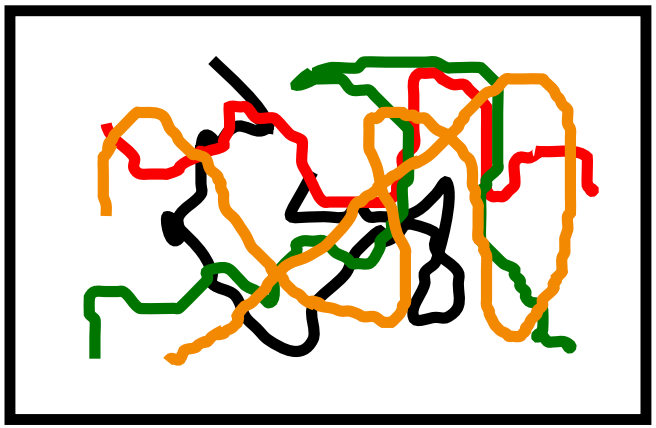
Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020

Massive parallel updates



One worker



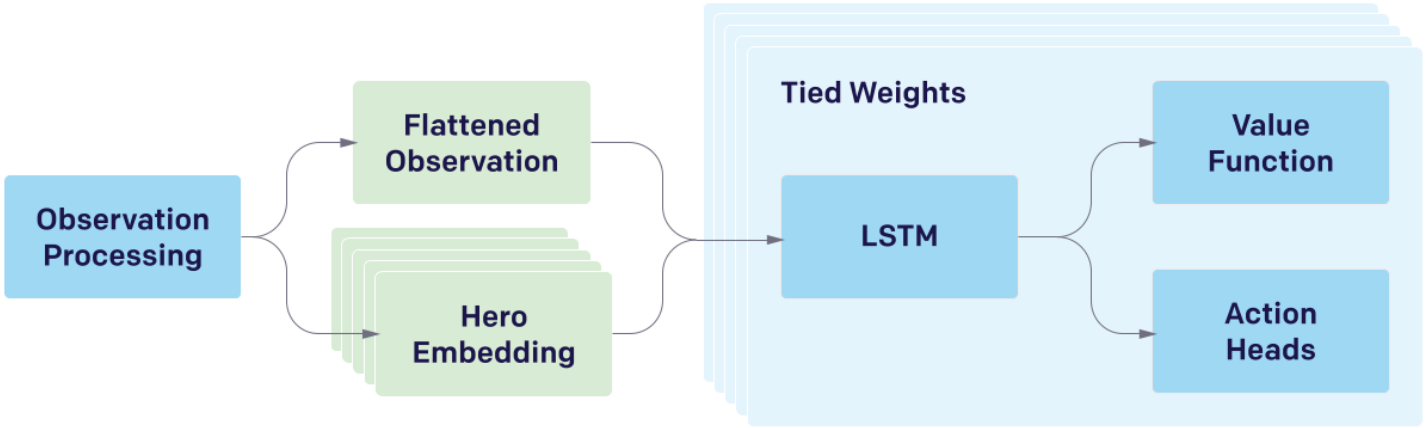
Many workers

- Several workers in parallel: more i.i.d and faster exploration
- The acceleration is better than linear in the number of workers
- No need for a replay buffer (as in A3C), but loss of sample efficiency

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al. (2018) Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*

Adamski, I., Adamski, R., Grel, T., Jedrych, A., Kaczmarek, K., & Michalewski, H. (2018) Distributed deep reinforcement learning: Learn how to play atari games in 21 minutes. *arXiv preprint arXiv:1801.02852*

OpenIA five



- The LSTM deals with non-Markov data
- The vision layers are problem specific

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019

Any question?

Send mail to: Olivier.Sigaud@upmc.fr


Adamski, I., Adamski, R., Grel, T., Jedrych, A., Kaczmarek, K., & Michalewski, H. (2018).
Distributed deep reinforcement learning: Learn how to play atari games in 21 minutes.
arXiv preprint arXiv:1801.02852.


Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., et al. (2019).
Solving rubik's cube with a robot hand.
arXiv preprint arXiv:1910.07113.


Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. (2019).
Dota 2 with large scale deep reinforcement learning.
arXiv preprint arXiv:1912.06680.


Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., & Kavukcuoglu, K. (2018).
IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures.
In J. G. Dy & A. Krause (Eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research* (pp. 1406–1415).: PMLR.


Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. (2018).
Soft actor-critic algorithms and applications.
arXiv preprint arXiv:1812.05905.


Heess, N., Sriram, S., Lemmon, J., Merel, J., Wayne, G., Tassa, Y., Erez, T., Wang, Z., Eslami, A., Riedmiller, M., et al. (2017).
Emergence of locomotion behaviours in rich environments.
arXiv preprint arXiv:1707.02286.


Huang, S., Kanervisto, A., Raffin, A., Wang, W., Ontañón, S., & Dossa, R. F. J. (2022).
A2C is a special case of PPO.
arXiv preprint arXiv:2205.09123.


Kakade, S. & Langford, J. (2002).

Approximately optimal approximate reinforcement learning.
In *ICML*, volume 2 (pp. 267–274).

Kakade, S. M. (2001).

A natural policy gradient.
In T. G. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]* (pp. 1531–1538).: MIT Press.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., & Kavukcuoglu, K. (2016).

Asynchronous methods for deep reinforcement learning.
In M. Balcan & K. Q. Weinberger (Eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings* (pp. 1928–1937).: JMLR.org.

OpenAI, Andrychowicz, M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., et al. (2020).
Learning dexterous in-hand manipulation.
The International Journal of Robotics Research, 39(1), 3–20.

Peters, J., Mülling, K., & Altun, Y. (2010).

Relative entropy policy search.
In *AAAI* (pp. 1607–1612).: Atlanta.

Peters, J. & Schaal, S. (2008).

Natural actor-critic.
Neurocomputing, 71(7-9), 1180–1190.

Pierrot, T., Perrin, N., & Sigaud, O. (2018).

First-order and second-order variants of the gradient descent: a unified framework.
arXiv preprint arXiv:1810.08102.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., & Moritz, P. (2015).

Trust region policy optimization.

In F. R. Bach & D. M. Blei (Eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings* (pp. 1889–1897).: JMLR.org.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017).

Proximal policy optimization algorithms.
arXiv preprint arXiv:1707.06347.

Wu, Y., Mansimov, E., Grosse, R. B., Liao, S., & Ba, J. (2017).

Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation.
In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA* (pp. 5279–5288).