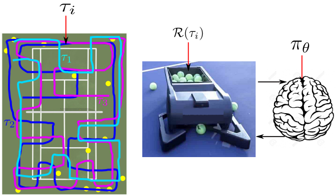
From Policy Gradient to Actor-Critic methods

The policy gradient derivation (1/3)

Olivier Sigaud

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

Reminder: policy search formalization



- τ_i is a robot trajectory
- $R(\tau_i)$ is the corresponding return
- $\pi_{\boldsymbol{\theta}}$ is the parametrized policy of the robot

- We want to optimize $J(\boldsymbol{\theta}) = \mathbb{E}_{\tau \sim \pi_{\boldsymbol{\theta}}}[R(\tau)]$, the global utility function
- We tune policy parameters $\boldsymbol{\theta}$, thus the goal is to find

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmax}}\, J(\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\text{argmax}} \sum_{\tau} P(\tau, \boldsymbol{\theta}) R(\tau) \tag{1}$$

- where $P(\tau, \boldsymbol{\theta})$ is the probability of trajectory τ under policy $\pi_{\boldsymbol{\theta}}$

Deisenroth, M. P., Neumann, G., Peters, J., et al. (2013) A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142

Policy Gradient approach

- General idea: increase $P(\tau, \boldsymbol{\theta})$ for trajectories τ with a high return
- Gradient ascent: Following the gradient from analytical knowledge
- Issue: in general, the function $J(\boldsymbol{\theta})$ is unknown
- How can we apply gradient ascent without knowing the function?
- The answer is the Policy Gradient Theorem

Policy Gradient approach (2)

- Direct policy search works with $< \boldsymbol{\theta}, J(\boldsymbol{\theta}) >$ samples
- It ignores that the return comes from state and action trajectories generated by a controller $\pi_{\boldsymbol{\theta}}$
- We can obtain explicit gradients by taking this information into account
- Not black-box anymore: access the state, action and reward at each step
- The transition and reward functions are still unknown (gray-box approach)
- Requires some math magics
- This lesson builds on "Deep RL bootcamp" youtube video $\#4A$: https://www.youtube.com/watch?v=S_gwYj1Q-44 (Pieter Abbeel)

Plain Policy Gradient (step 1)

- We are looking for $\boldsymbol{\theta}^* = \text{argmax}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \text{argmax}_{\boldsymbol{\theta}} \sum_\tau P(\tau, \boldsymbol{\theta}) R(\tau)$

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \sum_\tau P(\tau, \boldsymbol{\theta}) R(\tau) \\
&= \sum_\tau \nabla_{\boldsymbol{\theta}} P(\tau, \boldsymbol{\theta}) R(\tau) && \text{* gradient of sum is sum of gradients} \\
&= \sum_\tau \frac{P(\tau, \boldsymbol{\theta})}{P(\tau, \boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}} P(\tau, \boldsymbol{\theta}) R(\tau) && \text{* Multiply by one} \\
&= \sum_\tau P(\tau, \boldsymbol{\theta}) \frac{\nabla_{\boldsymbol{\theta}} P(\tau, \boldsymbol{\theta})}{P(\tau, \boldsymbol{\theta})} R(\tau) && \text{* Move one term} \\
&= \sum_\tau P(\tau, \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log P(\tau, \boldsymbol{\theta}) R(\tau) && \text{* by property of gradient of log} \\
&= \mathbb{E}_\tau [\nabla_{\boldsymbol{\theta}} \log P(\tau, \boldsymbol{\theta}) R(\tau)] && \text{* by definition of the expectation}
\end{aligned}
$$

Plain Policy Gradient (step 2)

- We want to compute $\mathbb{E}_\tau[\nabla_{\boldsymbol{\theta}} \log P(\tau, \boldsymbol{\theta}) R(\tau)]$
- We do not have an analytical expression for $P(\tau, \boldsymbol{\theta})$
- Thus the gradient $\nabla_{\boldsymbol{\theta}} \log P(\tau, \boldsymbol{\theta}) R(\tau)$ cannot be computed
- Let us reformulate $P(\tau, \boldsymbol{\theta})$ using the policy $\pi_{\boldsymbol{\theta}}$
- What is the probability of a trajectory?
- At each step, probability of taking each action (defined by the policy) times probability of reaching the next state given the action
- Then product over states for the whole horizon H

$$P(\tau, \boldsymbol{\theta}) = \prod_{t=1}^{H} p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t).\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t) \tag{2}$$

- (Strong) Markov assumption here: holds if steps are independent

Plain Policy Gradient (step 2 continued)

- Thus, under Markov assumption,

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} \log \mathrm{P}(\tau, \boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \log [\prod_{t=1}^{H} p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t).\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)] \\
& \quad \text{* log of product is sum of logs} \\
&= \nabla_{\boldsymbol{\theta}} [\sum_{t=1}^{H} \log\ \mathrm{p}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) + \sum_{t=1}^{H} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)] \\
&= \nabla_{\boldsymbol{\theta}} \sum_{t=1}^{H} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t) \ \text{* because first term independent of } \boldsymbol{\theta} \\
&= \sum_{t=1}^{H} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t) \ \text{* no dynamics model required!}
\end{aligned}
$$

- The key is here: we know $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)$!

Plain Policy Gradient (step 2 continued)

- The expectation $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_\tau[\nabla_{\boldsymbol{\theta}} \log P(\tau, \boldsymbol{\theta}) R(\tau)]$ can be rewritten

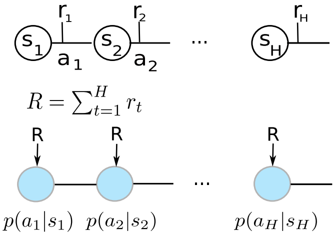
$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_\tau[\sum_{t=1}^{H} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t) R(\tau)]$$

- The expectation can be approximated by sampling over m trajectories:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{H} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t^{(i)}|\mathbf{s}_t^{(i)}) R(\tau^{(i)}) \tag{3}$$

- The policy structure $\pi_{\boldsymbol{\theta}}$ is known, thus the gradient $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s})$ can be computed for any pair $(\mathbf{s}, \mathbf{a})$
- We moved from direct policy search on $J(\boldsymbol{\theta})$ to gradient ascent on $\pi_{\boldsymbol{\theta}}$
- Can be turned into a practical (but not so efficient) algorithm

Algorithm 1

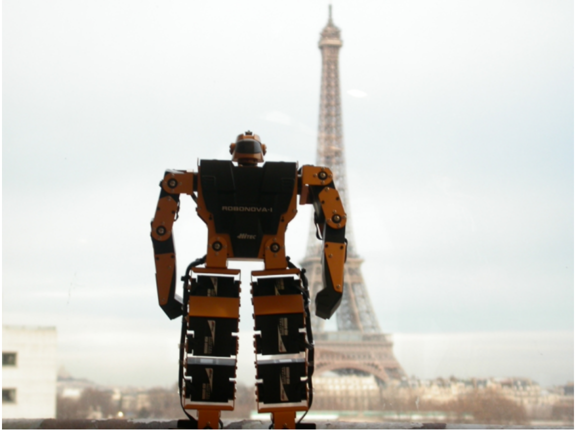


- Sample a set of trajectories from $\pi_{\boldsymbol{\theta}}$
- Compute:

$$Loss(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{H} \log\pi_{\boldsymbol{\theta}}(\mathbf{a}_t^{(i)}|\mathbf{s}_t^{(i)}) R(\tau^{(i)}) \tag{4}$$

- Minimize the loss using the NN backprop function with your favorite pytorch or tensorflow optimizer (Adam, RMSProp, SGD...)
- Iterate: sample again, for many time steps
- Note: if $R(\tau) = 0$, does nothing
- Next lesson: Policy gradient improvement

Any question?



Send mail to: `Olivier.Sigaud@upmc.fr`

Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al.
A survey on policy search for robotics.
Foundations and Trends® in Robotics, 2(1–2):1–142, 2013.