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

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

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

$$\theta^* = \arg\max_{\theta} J(\theta) = \arg\max_{\theta} \sum_{\tau} P(\tau|\theta)R(\tau) \quad (1)$$

- where $P(\tau|\theta)$ is the probability of trajectory $\tau$ under policy $\pi_\theta$

Policy Gradient approach

- General idea: increase $P(\tau|\theta)$ for trajectories $\tau$ with a high return
- Gradient ascent: Following the gradient from analytical knowledge
- Issue: in general, the function $J(\theta)$ 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 $< \theta, J(\theta)>$ samples
▶ It ignores that the return comes from state and action trajectories generated by a controller $\pi_\theta$
▶ 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 $\theta^* = \arg\max_\theta J(\theta) = \arg\max_\theta \sum_\tau P(\tau|\theta)R(\tau)$

$$\nabla_\theta J(\theta) = \nabla_\theta \sum_\tau P(\tau|\theta)R(\tau)$$

$$= \sum_\tau \nabla_\theta P(\tau|\theta)R(\tau) \quad \text{* gradient of sum is sum of gradients}$$

$$= \sum_\tau \frac{P(\tau|\theta)}{P(\tau|\theta)} \nabla_\theta P(\tau|\theta)R(\tau) \quad \text{* Multiply by one}$$

$$= \sum_\tau P(\tau|\theta) \frac{\nabla_\theta P(\tau|\theta)}{P(\tau|\theta)} R(\tau) \quad \text{* Move one term}$$

$$= \sum_\tau P(\tau|\theta) \nabla_\theta \log P(\tau|\theta) R(\tau) \quad \text{* by property of gradient of log}$$

$$= \mathbb{E}_\tau[\nabla_\theta \log P(\tau|\theta) R(\tau)] \quad \text{* by definition of the expectation}$$
Plain Policy Gradient (step 2)

- We want to compute $\mathbb{E}_\tau[\nabla_\theta \log P(\tau|\theta)R(\tau)]$
- We do not have an analytical expression for $P(\tau|\theta)$
- Thus the gradient $\nabla_\theta \log P(\tau|\theta)R(\tau)$ cannot be computed
- Let us reformulate $P(\tau|\theta)$ using the policy $\pi_\theta$
- 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 $H$

$$P(\tau|\theta) = \prod_{t=1}^{H} p(s_{t+1}|s_t, a_t) \cdot \pi_\theta(a_t|s_t)$$  \hspace{1cm} (2)

- (Strong) Markov assumption here: holds if steps are independent
Plain Policy Gradient (step 2 continued)

Thus, under Markov assumption,

\[ \nabla_{\theta} \log P(\tau, \theta) = \nabla_{\theta} \log \left[ \prod_{t=1}^{H} p(s_{t+1} | s_t, a_t) \pi_{\theta}(a_t | s_t) \right] \]

* log of product is sum of logs

\[ = \nabla_{\theta} \left[ \sum_{t=1}^{H} \log p(s_{t+1} | s_t, a_t) + \sum_{t=1}^{H} \log \pi_{\theta}(a_t | s_t) \right] \]

\[ = \nabla_{\theta} \sum_{t=1}^{H} \log \pi_{\theta}(a_t | s_t) \quad \text{* because first term independent of } \theta \]

\[ = \sum_{t=1}^{H} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \quad \text{* no dynamics model required!} \]

The key is here: we know \( \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \)!
Plain Policy Gradient (step 2 continued)

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

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau [\sum_{t=1}^{H} \nabla_\theta \log \pi_\theta (a_t|s_t) R(\tau)]$$

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

$$\nabla_\theta J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{H} \nabla_\theta \log \pi_\theta (a_t^{(i)}|s_t^{(i)}) R(\tau^{(i)})$$  \hspace{1cm} (3)

- The policy structure $\pi_\theta$ is known, thus the gradient $\nabla_\theta \log \pi_\theta (a|s)$ can be computed for any pair $(s, a)$

- We moved from direct policy search on $J(\theta)$ to gradient ascent on $\pi_\theta$

- Can be turned into a practical (but not so efficient) algorithm
Algorithm 1

Sample a set of trajectories from $\pi_\theta$
Compute:

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

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.