

From Policy Gradient to Actor-Critic methods

The policy gradient derivation (3/3)

Olivier Sigaud

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



Policy Gradient with constant baseline

► Reminder:

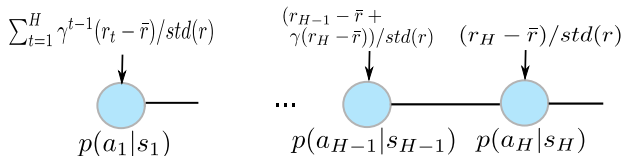
$$\nabla_{\theta} J(\theta) = \frac{1}{m} \sum_{i=1}^m \sum_{t=1}^H \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t^{(i)} | \mathbf{s}_t^{(i)}) \left[\sum_{k=t}^H \gamma^{k-t} r(\mathbf{s}_k^{(i)}, \mathbf{a}_k^{(i)}) \right] \quad (1)$$

- If all rewards are positive, the gradient increases all probabilities
- But with renormalization, only the largest increases emerge
- We can subtract a “baseline” to (1) without changing its mean:

$$\nabla_{\theta} J(\theta) = \frac{1}{m} \sum_{i=1}^m \sum_{t=1}^H \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t^{(i)} | \mathbf{s}_t^{(i)}) \left[\sum_{k=t}^H \gamma^{k-t} r(\mathbf{s}_k^{(i)}, \mathbf{a}_k^{(i)}) - b \right]$$

- A first baseline is the average return \bar{r} over all states of the batch
- Intuition: returns greater than average get positive, smaller get negative
- Use $(r_t^{(i)} - \bar{r})$ and divide by std → get a mean = 0 and a std = 1
- This improves variance (does the job of renormalization)
- Suggested in <https://www.youtube.com/watch?v=tqrcjHuNdmQ>

Algorithm 4: adding a constant baseline



- ▶ Estimate \bar{r} and $std(r)$ from all rollouts
- ▶ Same as Algorithm 2, using $(r_t^{(i)} - \bar{r}) / std(r)$
- ▶ Suffers from even less variance
- ▶ Does not work if all rewards r are identical (e.g. CartPole)

Policy Gradient with state-dependent baseline

- ▶ No impact on the gradient as long as the baseline does not depend on action
- ▶ A better baseline is $b(s_t) = V^\pi(s_t) = \mathbb{E}_\tau[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots + \gamma^{H-t} r_H]$
- ▶ The expectation can be approximated from the batch of trajectories
- ▶ Thus we get

$$\nabla_{\theta} J(\theta) = \frac{1}{m} \sum_{i=1}^m \sum_{t=1}^H \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t^{(i)} | \mathbf{s}_t^{(i)}) [Q^{\pi_{\theta}}(\mathbf{s}_t^{(i)} | \mathbf{a}_t^{(i)}) - V^{\pi_{\theta}}(\mathbf{s}_t^{(i)})]$$

- ▶ $A^\pi(s_t, \mathbf{a}_t) = Q^\pi(s_t | \mathbf{a}_t) - V^\pi(s_t)$ is the advantage function
- ▶ And we get

$$\nabla_{\theta} J(\theta) = \frac{1}{m} \sum_{i=1}^m \sum_{t=1}^H \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t^{(i)} | \mathbf{s}_t^{(i)}) A^{\pi_{\theta}}(\mathbf{s}_t^{(i)}, \mathbf{a}_t^{(i)})$$

https://www.youtube.com/watch?v=S_gwYj1Q-44 (27')



Williams, R. J. (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256



Estimating $V^\pi(s)$

- ▶ As for estimating $Q^\pi(s, a)$, but simpler
- ▶ Two approaches:
 - ▶ **Monte Carlo estimate:** Regression against empirical return

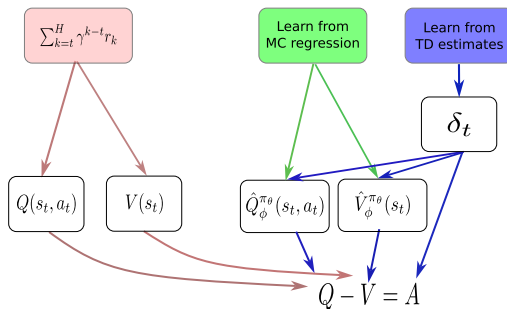
$$\phi_{j+1} \rightarrow \arg \min_{\phi_j} \frac{1}{m} \sum_{i=1}^m \sum_{t=1}^H ((\sum_{k=t}^H \gamma^{k-t} r(\mathbf{s}_k^{(i)}, \mathbf{a}_k^{(i)})) - \hat{V}_{\phi_j}^\pi(\mathbf{s}_t^{(i)}))^2$$

- ▶ **Temporal Difference estimate:** init $\hat{V}_{\phi_0}^\pi$ and fit using $(\mathbf{s}, \mathbf{a}, r, \mathbf{s}')$ data

$$\phi_{j+1} \rightarrow \arg \min_{\phi_j} \sum_{(\mathbf{s}, \mathbf{a}, r, \mathbf{s}')} ||r + \gamma \hat{V}_{\phi_j}^\pi(\mathbf{s}') - \hat{V}_{\phi_j}^\pi(\mathbf{s})||^2$$

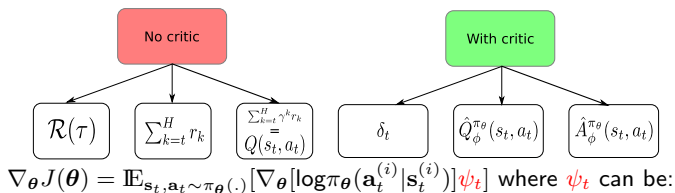
- ▶ May need some regularization to prevent large steps in ϕ

Different ways to compute the advantage



- $\hat{A}_\phi^{\pi_\theta}(s_t, \mathbf{a}_t) = \mathbb{E}[\delta_t]$
- Algorithm 5: Same as Algorithm 3 using $A^{\pi_\theta}(s_t^{(i)} | \mathbf{a}_t^{(i)})$ instead of $Q^{\pi_\theta}(s_t^{(i)} | \mathbf{a}_t^{(i)})$
- Suffers from even less variance
- Only with blue arrows we can perform updates based on the current state information

Synthesis



1. $\sum_{t=0}^H \gamma^t r_t$: total (discounted) reward of trajectory
2. $\sum_{k=t}^H \gamma^{k-t} r_k$: sum of rewards after \mathbf{a}_t
3. $\sum_{k=t}^H \gamma^{k-t} r_k - b(s_t)$: sum of rewards after \mathbf{a}_t with baseline
4. $\delta_t = r_t + \gamma V^{\pi}(\mathbf{s}_{t+1}) - V^{\pi}(\mathbf{s}_t)$: TD error
5. $\hat{Q}_{\phi}^{\pi_{\theta}}(\mathbf{s}_t, \mathbf{a}_t)$: action-value function
6. $\hat{A}_{\phi}^{\pi_{\theta}}(\mathbf{s}_t, \mathbf{a}_t)$: advantage function (the most used)

► Next lesson: Difference between PG with baseline and Actor-Critic



John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015

Any question?



Send mail to: Olivier.Sigaud@upmc.fr