# Actor-Critics

Christopher Mutschler

Fraunhofer IIS
FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

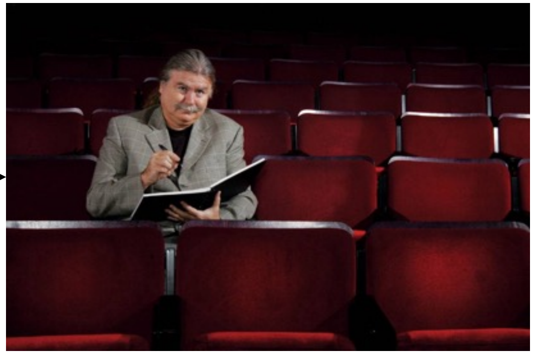
# Policy Gradients: Variance (revisited)

$$\nabla_\theta \mathbb{E}_{\pi_\theta} G(\tau) \approx \frac{1}{L} \sum_{\tau} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \underbrace{\sum_{t'=t}^{T} \gamma^{t'-t} R(s_{t'}, a_{t'})}_{= \, Q^\pi(s_t, a_t)} - b(s_t)$$

- Monte-Carlo policy gradient is sampled and has high variance
- Idea: we can use a critic that estimates the Q

actor

critic



Fraunhofer IIS | FAU Friedrich-Alexander Universität Erlangen-Nürnberg, Technische Fakultät

# Policy Gradients: Variance (revisited)

$$\nabla_\theta \mathbb{E}_{\pi_\theta} G(\tau) \approx \frac{1}{L} \sum_{\tau} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \underbrace{\sum_{t'=t}^{T} \gamma^{t'-t} R(s_{t'}, a_{t'}) - b(s_t)}_{= \; Q^\pi(s_t, a_t)}$$

- Monte-Carlo policy gradient is sampled and has high variance
- Idea: we can use a critic that estimates the Q

→ In practice, (as we already know) $Q^\pi(s_t, a_t)$ cannot be "computed"

→ we instead need to approximate it with a neural network with parameters $\phi$ and standard SGD over $k$ epochs minimizing the MSE:

$$\phi_k = \arg\min_{\phi} \mathbb{E}_{s_t; a_t, \hat{R}_t \sim \pi_k} \left[ \left( Q_\phi(s_t, a_t) - \hat{R}_t \right)^2 \right]$$

# Actor-Critic

$$\nabla_\theta \mathbb{E}_{\pi_\theta} G(\tau) \approx \frac{1}{L} \sum_\tau \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \underbrace{\sum_{t'=t}^{T} \gamma^{t'-t} R(s_{t'}, a_{t'})}_{= \, Q^\pi(s_t, a_t)} - b(s_t)$$

- Monte-Carlo policy gradient is sampled and has high variance
- Idea: we can use a critic (e.g., a NN as in DQN) to estimate the Q and learn two sets of parameters separately
  - Actor: update $\theta$ by policy gradient
  - Critic: Update parameters $\phi$ of $v_\phi^\pi$, ? e.g., by n-step TD
- We call such algorithms *actor-critic algorithms*

# Actor-Critic

- Typically, we estimate $v^{\pi}(s_t; \phi)$ explicitly, and then sample

$$q^{\pi}(s_t, a_t) \approx G_t^{(n)}$$

- For instance: $\hat{G}_t^{(1)} = R_t + \gamma v^{\pi}(s_{t+1}; \phi)$

- Then we arrive at:

$$\nabla_\theta \mathbb{E}_{\pi_\theta} \hat{G}(\tau) = \frac{1}{L} \sum_{\tau} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t | s_t) \left(\hat{G}_t - b(s_t)\right)$$

→ We use an *approximate* gradient in the direction suggested by the critic

# Actor-Critic: Advantage Functions

$$\nabla_\theta \mathbb{E}_{\pi_\theta} G(\tau) \approx \frac{1}{L} \sum_{\tau} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \underbrace{(\hat{G}_t - b(s_t))}_{:= A^\pi(s_t, a_t)}$$

Calculating the advantage function is (embarrassingly) straight forward:

- As before, we can simply use the TD error:

$$\begin{aligned} A^\pi(s_t, a_t) &= Q^\pi(s_t, a_t) - V^\pi(s_t) \\ &= r + \gamma \cdot v^\pi(s_{t+1}) - v^\pi(s_t) \end{aligned}$$

# Actor-Critic: Advantage Functions

$$\nabla_\theta \mathbb{E}_{\pi_\theta} G(\tau) \approx \frac{1}{L} \sum_{\tau} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \underbrace{(\hat{G}_t - b(s_t))}_{:= A^\pi(s_t, a_t)}$$

Calculating the advantage function is (embarrassingly) straight forward:

- We can also use **Generalized Advantage Estimation (GAE)** (multi-step TD-error like TD with n-step returns)

$$\begin{aligned}
\hat{A}_t^{(1)} &:= \delta_t^V && = -V(s_t) + r_t + \gamma V(s_{t+1}) \\
\hat{A}_t^{(2)} &:= \delta_t^V + \gamma \delta_{t+1}^V && = -V(s_t) + r_t + \gamma V(s_{t+1}) + \gamma^2 V(s_{t+2}) \\
\hat{A}_t^{(3)} &:= \delta_t^V + \gamma \delta_{t+1}^V + \gamma^2 \delta_{t+2}^V && = -V(s_t) + r_t + \gamma V(s_{t+1}) + \gamma^2 V(s_{t+2}) + \gamma^3 V(s_{t+3}) \\
&\dots \\
\hat{A}_t^{(k)} &:= \textstyle\sum_{l=0}^{k-1} \gamma^l \delta_{t+l}^V && = -V(s_t) + r_t + \gamma r_{t+1} + \dots + \gamma^{k-1} r_{t+k-1} + \gamma^k V(s_{t+k})
\end{aligned}$$

$$\hat{A}_t^{(\infty)} = \sum_{l=0}^{\infty} \gamma^l \delta_{t+l}^V = -V(s_t) + \sum_{l=0}^{\infty} \gamma^l + r_{t+l}$$

# Actor-Critic: Advantage Functions

$$\nabla_\theta \mathbb{E}_{\pi_\theta} G(\tau) \approx \frac{1}{L} \sum_{\tau} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t | s_t) \underbrace{(\hat{G}_t - b(s_t))}_{:= A^\pi(s_t, a_t)}$$

Calculating the advantage function is (embarrassingly) straight forward:

- Full returns: high variance
- One-step TD: high bias
- n-step TD: somewhere in the middle

But still: approximating the policy gradient introduces bias

- It is important to use on-policy targets (i.e., can be corrected using importance sampling)
- Alternative idea: bootstrap (with $\lambda = 0$) whenever policies differ

# Continuous Actions

- Because we directly update the policy parameters of the policy, we can easily deal with continuous action spaces
- Most algorithms discussed can be used for both discrete and continuous actions
- However: exploration in high-dimensional continuous spaces might be challenging

# Continuous Actions: Gaussian Policy

- In continuous action spaces, a Gaussian policy is common, e.g., mean is some function of state $\mu(s)$
- For simplicity, lets consider fixed variance of $\sigma^2$ (which can be parameterized as well, instead)
- Policy is Gaussian: $a \sim \mathcal{N}(\mu(s), \sigma^2)$
- The gradient of the log of the policy is then

$$\nabla_\theta \log \pi_\theta(s, a) = \frac{a - \mu(s)}{\sigma^2} \nabla \mu(s)$$

- This can be used, for instance, in REINFORCE or advantage actor-critic

# Conclusion

- The policy gradient has many forms:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) G_t]$$ REINFORCE

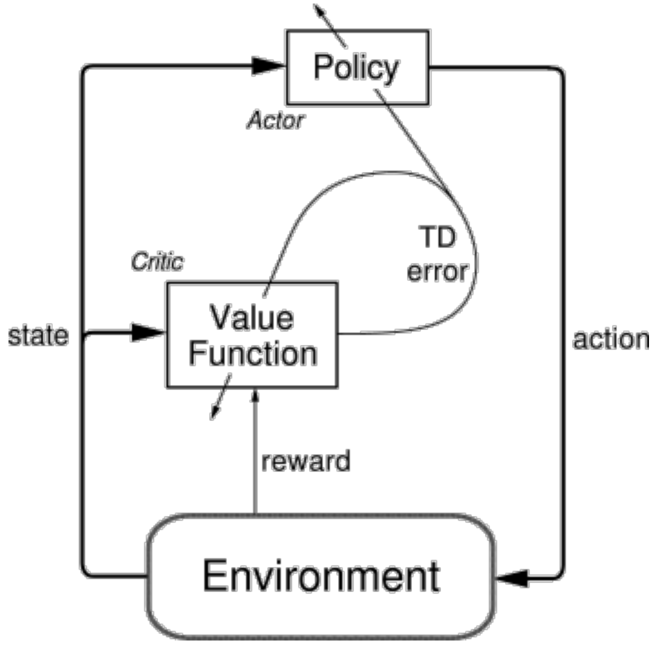
$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) (G_t - b(s_t))]$$ REINFORCE w/ baseline

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) \hat{A}_t^{(\infty)}\right]$$ advantage actor-critic

$$\nabla_\theta J(\theta) = \nabla_\theta Q_t(s, \pi_\theta(s))$$ deterministic policy gradient (see video on DDPG)

- Each leads to a stochastic gradient ascent algorithm

- Critic uses policy evaluation (e.g., MC or TD) to estimate $Q^\pi(s,a)$ or $V^\pi(s)$

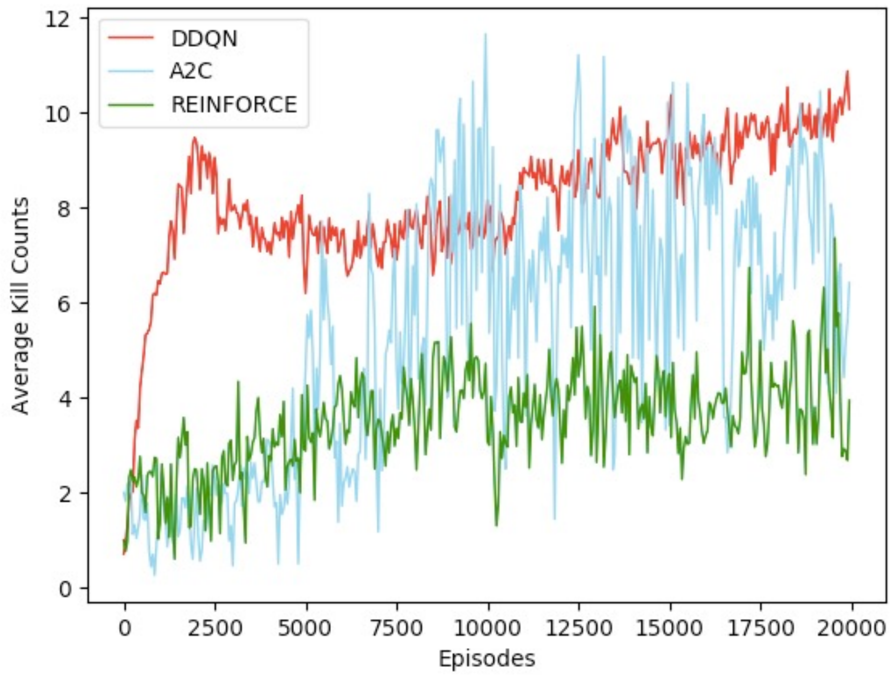



*Sutton et a.: Reinforcement learning: An introduction. 2018.*

# Conclusion (cont'd)

- It is substantially different from DQNs
  - no replay buffer, no stored experiences
  - learn directly, on-policy

- Once a batch has been used → discard experience
  - less sample efficient




*https://flyyufelix.github.io/2017/10/12/dqn-vs-pg.html*

- Learning off-policy (which allows to reuse experience) is not (or only with certain tricks) possible

# One more thing: Exploration-Exploitation

- Policy-gradients only consider improvement under current data
  → easy to get stuck in local optima
- Could we use $\epsilon$-greedy? – Yes! but it is not ideal…
  - Wildly different actions cause breakage
  - Exploration is mostly uninformed about current best guess

- Alternative idea: make sure that the entropy of the policy is not too low:

$$-\sum_s \mu(s) \sum_a \pi(a|s) \log \pi(a|s) = -\mathbb{E}[\log \pi(a_t|s_t)]$$

→Add a regularization term that pushes entropy up slightly each step

- Encourages exploration and does not pick fully randomly
  - May increase variance in Gaussian policies
  - Makes softmax slightly more uniform
- Similar to (spoiler!) KL-regularization (in TRPO)
- Works well in practice

# Lessons Learned

*"Always try to solve a problem the direct way*
*– but be careful with the high variance."*

*https://www.youtube.com/watch?v=ek3hgVQ1RqM*

# References


- Deep RL Bootcamp 2017:
  - Pieter Abbeel: Policy Gradients (Lecture 4A), Aug. 26th, 2017
  - Andrej Karpathy: Pong from Pixels (Lecture 4b), Aug. 26th, 2017
- https://www.janisklaise.com/post/rl-policy-gradients/
- OpenAI Spinning Up: “Vanilla Policy Gradient”. https://spinningup.openai.com/en/latest/algorithms/vpg.html
- OpenAI Spinning Up: “Intro to Policy Optimization”. https://spinningup.openai.com/en/latest/spinningup/rl_intro3.html#other-forms-of-the-policy-gradient
- Williams, Ronald J.: Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. Machine Learning 8:229-256.