Deep Reinforcement Learning

- so far: discussed action-value methods for RL
  → policy defined from $v_\pi(s)$ or $q_\pi(s,a)$

$$v_\pi(s) \xrightarrow{DRL} v_\theta(s)$$
$$q_\pi(s,a) \longrightarrow q_\theta(s,a)$$
$$\pi(a|s) \longrightarrow \pi_\theta(s,a)$$

approximate value functions & policy using a ML model with params θ: weights & biases

Policy Gradient Methods

- algorithms that learn a parametrized policy directly: $\pi_\theta(s|a)$
  - value function may be present, but it is not required to select actions
    (→ actor-critic algorithms)

idea: parametrize policy by some variational parameters $\theta \in \mathbb{R}^d$

$$\pi(a|s) \approx \pi_\theta(a|s)$$

1) <u>which</u> <u>variational ansatz</u> /model do we use for $\theta \longmapsto \pi_\theta(a|s)$?

e.g. i) if action space A is <u>discrete</u>:

<u>softmax</u> <u>policy</u>:

$$\pi_\theta(a|s) := \frac{e^{\beta h_\theta(s,a)}}{\sum_{a \in A} e^{\beta h_\theta(s,a)}}$$

β: (inverse) temperature

- $\beta \to \infty$: policy is greedy
- $\beta \to 0$: equiprobable policy

$h_\theta(s,a)$: some parametrized function on $S \times A$, e.g. DNN or CNN or anything suitable to problem of study

e.g. Gridworlds, Atari games, board games

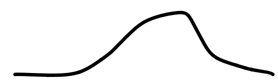
ii) <u>continuous</u> action space A

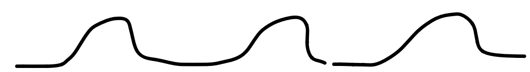
e.g. Gaussian policy:

$$\pi_\theta(a|s) := \frac{1}{\sqrt{2\pi\sigma_\theta^2(s)}} e^{-\frac{(a-\mu_\theta(s))^2}{2\sigma_\theta^2(s)}}$$

$\theta \mapsto \mu_\theta$ parametrize the mean μ_θ
$\theta \mapsto \sigma_\theta$ & variance σ^2_θ of the Gaussian policy

e.g. $s \longrightarrow DNN_\theta \begin{cases} \nearrow \mu_\theta \\ \searrow \sigma_\theta \end{cases} \Big\} \longrightarrow \pi_\theta$

<u>issue</u>: unimodality: 

opt. π_* 

→ single Gaussian insuff.

→ Gaussian mixture models, corrected Gaussian etc.

- <u>possible</u> <u>advantages</u> of policy gradient methods (over value-function methods):

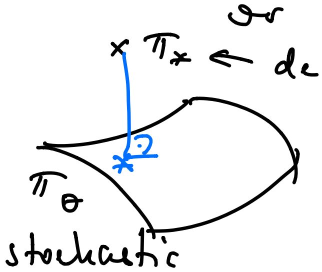
→ $\pi_\theta(a|s)$ can approach a deterministic policy (not possible if we use ε-greedy policies, e.g. SARSA, Q-learning etc.)

- use (inv.) temp parameter β to control stochasticity using a schedule:

$$\beta = \beta(n) \quad , n: \text{ episode number}$$

- using a schedule we can enforce convergence of the algorithm (but to what?!)

→ parametrization enables (in principle) an arbitrary policy (can have a stochastic optimal policy)
e.g. reward is uncertain
or env. is stochastic



→ learning the policy might just be easier than learning a value function!

→ can use policy to build in prior knowledge about problem (pretraining, etc.)

2) how do we find optimal parameters θ? s.t. $\pi_\theta(a|s) \approx \pi_*(a|s)$ for a given task?

→ assume that $\theta \mapsto \pi_\theta(a|s)$ is diff'ble

⇒ find those params θ which minimize expected return $\mathbb{E}_{\pi_\theta}[G_t \mid S_t = s_0]$

e.g. use gradient descent / ascent

Policy Gradient Theorem

- RL objective: total expected return starting from state s_0 & following policy π_θ afterwards

$$J(\theta) := \mathbb{E}_{\pi_\theta}[G_t \mid S_t = s_0]$$

$$= V_{\pi_\theta}(s_0) \quad \text{value function of } s_0$$

- compute gradient of the RL objective wrt. variational params. θ

$$\nabla_\theta v_{\pi_\theta}(s) = \nabla_\theta \sum_a \pi_\theta(a|s)\, q_{\pi_\theta}(s,a)$$

$$= \sum_a \left(\nabla_\theta \pi_\theta(a|s)\right) q_{\pi_\theta}(s,a) + \pi_\theta(a|s)\, \nabla_\theta\, q_{\pi_\theta}(s,a)$$

$$= \sum_a \Big\{ \left[\nabla_\theta \pi(a|s)\right] q_{\pi_\theta}(s,a) +$$

$$+ \pi_\theta(a|s)\, \nabla_\theta \sum_{s',r} p(s',r|s,a)\left[r + v_{\pi_\theta}(s')\right] \Big\}$$

indep. of θ

$\sum_r p(s',r|s,a) = p(s'|s,a)$

$$= \sum_a \Big\{ \left[\nabla_\theta \pi(a|s)\right] q_{\pi_\theta}(s,a) +$$

$$+ \pi_\theta(a|s) \sum_{s'} p(s'|s,a)\, \nabla_\theta v_{\pi_\theta}(s') \Big\}$$

unroll

$$= \sum_a \Big\{ \left[\nabla_\theta \pi_\theta(a|s)\right] q_{\pi_\theta}(s,a) +$$

$$\pi_\theta(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \Big[\left(\nabla_\theta \pi_\theta(a'|s')\right) q_{\pi_\theta}(s',a') +$$

$$+ \pi_\theta(a'|s') \sum_{s''} p(s''|s',a')\, \nabla_\theta v_{\pi_\theta}(s'') \Big] \Big\}$$

$$= \dots =$$

$$= \sum_{x \in S} \sum_{k=0}^{\infty} Pr_{\pi_\theta}(s \to x; k) \sum_a \left[\nabla_\theta \pi_\theta(a|x)\right] q_{\pi_\theta}(x,a)$$

where $Pr_{\pi_\theta}(s \to x; k)$ is the prob. to transition from $s \to x$ in k steps after following π_θ

→ evaluate on initial state s_0:

$$\nabla_\theta J(\theta) = \sum_{s\in S} \underbrace{\left(\sum_{k=0}^{\infty} Pr_{\pi_\theta}(s_0 \to s; k)\right)}_{=:\, \eta_{\pi_\theta}(s)} \sum_a \left(\nabla_\theta \pi_\theta(a|s)\right) q_{\pi_\theta}(s,a)$$

<u>issue</u>: $\eta_{\pi_\theta}(s)$ is <u>not</u> a prob. distr.
(not normalized)

$$\nabla_\theta J(\theta) = \underbrace{\sum_{s'} \eta_{\pi_\theta}(s')}_{\substack{=\text{const.}\\ \text{(indep. of } \theta)}} \sum_s \underbrace{\frac{\eta_{\pi_\theta}(s)}{\sum_{s''} \eta_{\pi_\theta}(s'')}}_{=:\, \mu_{\pi_\theta}(s) \text{ normalized!}} \sum_a \nabla_\theta \pi_\theta(a|s) \times q_{\pi_\theta}(s,a)$$

$$\nabla_\theta J(\theta) \propto \sum_s \mu_{\pi_\theta}(s) \sum_a \nabla_\theta \pi_\theta(a|s) \times q_{\pi_\theta}(s,a)$$

overall const in GD is irrelevant in practice
b/c it can be absorbed in the learning rate/
step size of GD.

$$\boxed{\nabla_\theta J(\theta) \propto \sum_s \mu_{\pi_\theta}(s) \sum_a \nabla_\theta \pi_\theta(a|s) \times q_{\pi_\theta}(s,a)}$$

– <u>problem</u>: $\mu_{\pi_\theta}(s)$ requires knowledge of
the transition prob. $p(s'|s,a)$
→ it requires a model

→ way out : MC methods:
need to write $\nabla_\theta J(\theta)$ as an expectation over trajectories $\tau = (S_0, A_0, S_1, A_1, \dots)$

$$\nabla_\theta J = \sum_{s,a} \mu_{\pi_\theta}(s)\, q_{\pi_\theta}(s,a)\, \pi_\theta(a|s) \underbrace{\frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)}}_{= \nabla_\theta \log \pi_\theta(a|s)}$$

$$= \sum_{s,a} \underbrace{\mu_{\pi_\theta}(s)\, \pi_\theta(a|s)}_{\substack{\text{probability for} \\ \text{pair } (s,a) \text{ to occur} \\ \text{under } \pi_\theta \text{ \& } p}} \nabla_\theta \log \pi_\theta(a|s) \times q_{\pi_\theta}(s,a)$$

-def : prob. of a trajectory τ to occur under policy π_θ & trans. prob. $p(s'|s,a)$

$$\underline{P}(\tau) := p(s_0)\, \pi_\theta(a_0|s_0)\, p(s_1|s_0,a_0)\, \pi_\theta(a_1|s_1)\, p(s_2|s_1,a_1) \dots$$
$$= p(s_0) \prod_{t=0}^{T-1} \pi_\theta(a_t|s_t)\, p(s_{t+1}|s_t,a_t)$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \underline{P}} \left[\nabla_\theta \log \pi(\tau) \times G(\tau) \right]$$

where $\pi_\theta(\tau) := \prod_{t=0}^{T-1} \pi_\theta(a_t|s_t)$; $G(\tau) = \sum_{t=0}^{T-1} r(s_t, a_t)$

$$\tau = (s_0, a_0, s_1, a_1, \dots, a_{T-1}, s_T)$$

- generate/sample N trajectories using MC from policy π_θ

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{j=1}^{N} \nabla_\theta \log \pi_\theta(\tau^j) \times G(\underbrace{\tau^j}_{\text{trajectory sampled using MC}})$$

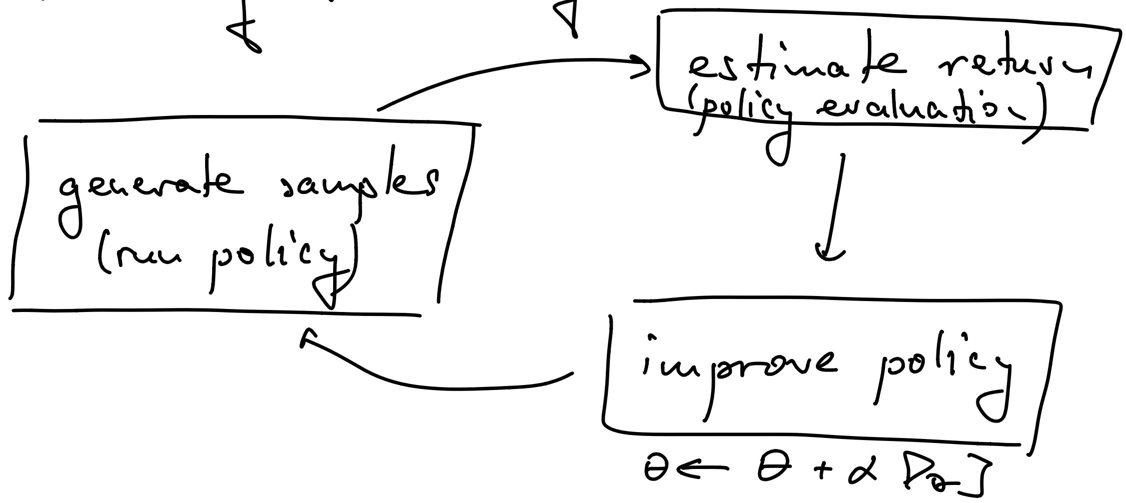
$$= \frac{1}{N} \sum_{j=1}^{N} \left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{\underbrace{t}_{\text{time step within the episode}}}^j \mid s_t^j) \right] \left[\sum_{t'=1}^{T} r(s_t^j, a_t^j) \right]$$

- update rule for parameters θ using SGA:

$$\theta \leftarrow \theta \underbrace{+}_{\text{ascent (maximize return)}} \underbrace{\alpha}_{\text{learning rate / step size}} \underbrace{\nabla_\theta J(\theta)}_{\text{use MC estimate}}$$

- alternative derivation in Berkeley lectures
- intuition about PG:
  - → PG : "gradient ascent for RL"
  - → PG makes higher-reward trajectories more likely
  - → PG : trial & error learning (→ MC)

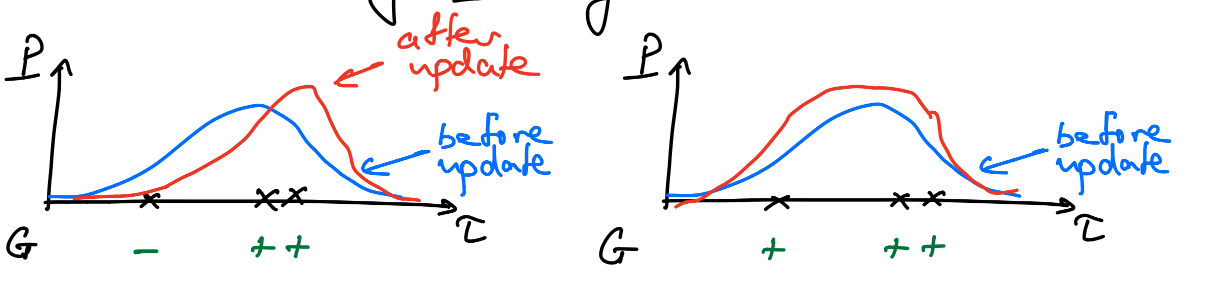
– Anatomy of RL algorithms



– REINFORCE algorithm for PG:

1. sample $\{\tau^j\}_{j=1}^{N}$ using MC from current policy π_θ (run policy)
2. estimate $\nabla_\theta J(\theta)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$ (improve policy)

– What can go wrong with REINFORCE?



$r \rightarrow r +$ large const.

$\rightarrow$ hints at a high variance problem

– how do we reduce the variance in PG?

1) take into account causality:

$$\nabla_\theta J \approx \frac{1}{N} \sum_{j=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta (a_t^j | s_t^j) \sum_{t'=1}^{T} r(s_{t'}^j, a_{t'}^j)$$

past actions affect future states & rewards but not vice-versa!

$\pi_{t'}$ cannot affect r_t at $t < t'$

$$\sum_{t'=1}^{T} r(s_{t'}^j, a_{t'}^j) \longrightarrow \sum_{t'=t}^{T} r(s_{t'}^j, a_{t'}^j) = q_t^j$$

reduces variance b/c we have to sum up fewer members

q_t^j : reward-to-go

2) baselines:

recall: i) PG makes higher reward trajectories more probable (intuition)

ii) e.g. $r \longrightarrow r + 10^9$ $\rightarrow$ all trajectories seem to be good want: make traj. which are better than average more likely & trajectories worse than average less likely.

$$b := \frac{1}{N} \sum_{j=1}^{N} r(\tau^j) \quad \text{average return}$$

want:

$$\nabla_\theta J \approx \frac{1}{N} \sum_j \nabla_\theta \log \pi_\theta(\tau^j) \left[G(\tau^j) - b \right]$$

issue: that's not the original gradient!

look at:

$$\mathbb{E}_{\tau \sim p}\left[b \nabla_\theta \log \pi_\theta(\tau) \right] = b\, \mathbb{E}_{\tau \sim p}\left[\nabla_\theta \log \pi_\theta(\tau) \right]$$

$$= b\, \mathbb{E}_{\tau \sim p}\left[\frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} \right]$$

$$= b \int d\tau\, \cancel{\pi_\theta(\tau)} \frac{\nabla_\theta \pi_\theta(\tau)}{\cancel{\pi_\theta(\tau)}} = b \int d\tau\, \nabla_\theta \pi_\theta(\tau)$$

$$= b \nabla_\theta \int d\tau\, \pi(\tau) = b \nabla_\theta \underbrace{\mathbb{E}_{\tau \sim p}[1]}_{=1} = 0$$

→ adding a baseline does not change the expectation of the gradient! (→ baselines are unbiased)

but it changes its variance

- improved PG with baseline: (lower variance)

$$\nabla_\theta J \approx \frac{1}{N} \sum_{j=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta (a_t^j | s_t^j) \sum_{t'=t}^{T} \left(r(s_{t'}^j, a_{t'}^j) - b \right)$$

- optimal baseline for minimal variance:

recall :

$$\mathrm{Var}[X] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$$

$$\mathrm{Var}(b) = \mathbb{E}_{\tau \sim p} \left[(\nabla_\theta \log \pi_\theta (G-b))^2 \right] - \underbrace{\mathbb{E}_{\tau \sim p} [\nabla_\theta \log \pi_\theta (G-b)]^2}_{= \mathbb{E}_{\tau \sim p} [\nabla_\theta \log \pi_\theta G]^2}$$

$\to$ indep. of b!

$$\frac{\partial}{\partial b} \mathrm{Var}(b) = \mathbb{E}_{\tau \sim p} \left[(\nabla_\theta \log \pi_\theta)^2 (-2G + 2b) \right]$$

$$= -2 \mathbb{E}_{\tau \sim p} \left[(\nabla_\theta \log \pi_\theta)^2 G \right] + 2b \, \mathbb{E}_{\tau \sim p} \left[(\nabla_\theta \log \pi_\theta)^2 \right]$$

$$\frac{\partial}{\partial b} \mathrm{Var}(b) \overset{!}{=} 0$$

$$\Rightarrow b_{\text{optimal}} = \frac{\mathbb{E}_{\tau \sim p} \left[(\nabla_\theta \log \pi_\theta)^2 G \right]}{\mathbb{E}_{\tau \sim p} \left[(\nabla_\theta \log \pi_\theta)^2 \right]}$$

understand division element-wise

$\to$ b_{optimal} knows about different directions in $J(\theta)$-landscape

→ drawback : not really used in practice
b/c tedious to implement

— Practical implementation of PG :

→ how do we efficiently compute $\nabla_\theta J$?

$$\nabla_\theta J \approx \frac{1}{N} \sum_j \left[\nabla_\theta \log \pi_\theta(\tau^j)\right]\left(G(\tau^j) - b\right)$$

but : $J \approx \frac{1}{N} \sum_j G(\tau^j)$ indep. of θ
→ cannot use that as a cost function?

use a pseudo cost fn :
→ a cost/loss function, s.t.

$$\nabla_\theta J_{pseudo}(\theta) = \nabla_\theta J_{exact}(\theta)$$

but $J_{pseudo}(\theta) \neq J_{exact}(\theta)$

$$J_{pseudo}(\theta) \approx \frac{1}{N} \sum_j \log \pi_\theta(\tau^j) \left[G(\tau^j) - b\right]$$

— a few implementation tips :

1) use larger batches of data as compared to supervised learning (need to reduce variance of $\nabla_\theta J$)

2) learning rate of ADAM/SGD may need fine-tuning