# Origins of Actor-Critic

## Policy Gradients

$$\hookrightarrow \; E_{\tau \sim \pi_\theta(\tau)} \left[ \nabla_\theta \log \pi_\theta(\tau) \, A(\tau) \right]$$

$$= E_{\tau \sim \pi_\theta(\tau)} \left[ \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t | s_t) \times A(\tau) \right]$$

$$= \sum_{t=1}^{T} E_{a,s \sim \pi_\theta} \nabla_\theta \log \pi_\theta(a_t | s_t) \times A(\tau)$$

Now, if this was a CONSTANT,

Then $\nabla_\theta \log \pi_\theta(a_t | s_t)$ can be taken COMMON, across all the different trajectories!

So,

$$\sum_{t=1}^{T} \sum_{a,s} p_\theta(a,s) \nabla_\theta \log \pi_\theta(a|s) (K)$$

And this would be the marginal distribution probability!

But, Now, Because "K" is not same for all the trajectories, the grad, can't be GROUPED, taken COMMON to get the

So,

marginal probability in front!

Instead,

say, $P_\theta(a,s) = \frac{l}{N}$ → subclass of trajectories that all through different routes landed on the same $(a,s)$

$N$ → Total number of trajectories,

$P_\theta(a,s)$ will be distributed, the following way

$$\sum_{t=1}^{T} \sum_{\text{all } a,s} \nabla_\theta \log \pi_\theta(a|s) \left[ l_1 \tilde{Q}_1 + l_2 \tilde{Q}_2 \cdots + l_n \tilde{Q}_n \right]$$

Since we are estimating the policy gradient through Monte-Carlo!

where $l = l_1 + l_2 \cdots + l_n$

So,

$$\sum_{t=1}^{T} \sum_{\text{all } a,s} \nabla_\theta \log \pi_\theta(a|s) \left[ \frac{l_1 \tilde{Q}_1 + l_2 \tilde{Q}_2 \cdots + l_n \tilde{Q}_n}{N} \right]$$

is the policy gradient,

where $\tilde{Q}_1, \tilde{Q}_2 \cdots$ are the different routes via which it can end in, starting from $a_t, s_t$ & $l_1, l_2 \cdots$ are the frequencies for each route!

Now, Instead of dividing by N, if you divide by "l",

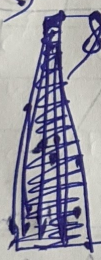
then it becomes (an) average of $Q$ estimates

$$\theta_{t+1} = \theta_t + \sum_{t=1}^{T} \sum_{\text{all } a,s} \nabla_\theta \log \pi_\theta(a|s) \left[ \frac{k_1 \tilde{Q}_1 + k_2 \tilde{Q}_2 \cdots k_n \tilde{Q}_n}{l} \right]$$

This is averaging of $Q$ estimates

$$\theta_{t+1} = \theta_t + \sum_{t=1}^{T} \sum_{\text{all } a,s} \nabla_\theta \log \pi_\theta(a|s) \, Q(a,s)$$

Actor-Critic (is) born!



$\theta_t$ doing this, changes the gradient, it introduces a "bias"!

Also, $l < N$, so the gradients are now being divided by a smaller no. so the gradients will be BIGGER!

But, In actor-critic, there is NO expectation anymore!

"biased"

Since, Actor-Critic is the RESULT of an expectation!

↳ So, in Actor-Critic, there is ONLY a sum!

So, if we know the Q-values perfectly somehow!

↳ Then, we don't need LOTS of sampling where $N \to \infty$

We just sample enough to get a wide variety of states & actions, to sum over gradients to $\theta$ from all those place & update our network!

A minibatch

↳ Smaller the batch, Higher the Thrashing & Oscillations!

So, ① Expectation in Policy Gradients gives rise to some sort of an averaging over $Q$ estimates

↳ We replace it by Q value taking in a little bit BIAS just like "Causality"

② Since the expectation is consumed in Q-values, there is NO expectation in actor, ONLY SUMMATION, just generate a nice variety, a minibatch

Now, the CRITIC.

Only for the 1st step,

(1) we generate [LOT] of samples ($N \to \infty$) like policy gradients & train till convergence, To make the critic clean Q-function!

(>1) from, next step on, we just generate a mini-batch, (for variety), for updating the actor! ~~and use~~ [every step]!

And since the policy is (just) changing a little bit @ each gradient step, we use the (same) minibatch to take (one) gradient step in "our" critic also, to update it for the (next) policy! Since the Q-function is also not changing much @ every step!

So, @ every step

- Sample a minibatch
- Update Critic to reflect ~~our~~ current policy
- Update current policy actor using critic to next policy

# Training (Critic)

Critic learns the V function and NOT the Q function.

This is because V fn is a function of the state $V(s)$ while Q fn is a function of both the state and the action $Q(s,a)$

So, if action is several dimensions, then the network grows! & there are a lot of parameters to train

So, $V(s)$ is learnt by critic to reduce network size!

And Q is approximated using V!

$$Q(s_t, a) = E\, V(s_{t+1})$$

$s_{t+1}$

we instead take a single $V(s_{t+1})$ which is a single sample estimate of the expectation & approximate Q by it

$$Q(s_t, a) = V(s_{t+1})$$

found in the samp

## Monte-Carlo Estimates

We can use the estimates we use in Policy Gradients as our training targets for our critic!

$$V(s) = \sum_{t=t}^{T} r_t,$$

This is a single sample estimate of the actual expectation for V.

Now since, we have many samples, so many estimates, & since all of them are targets together for our network, you might think gradient descent, would settle @ the average of these targets to minimize loss which is nothing BUT the expectation of the estimates!

But this anyway Policy Gradients also does for us! There is nothing new!

So, what is the advantage of introducing an entire network to do the same averaging as that of PG?

The point of introducing a network for the same averaging is,

Practically when you sample, only the most probable states, will repeat a lot!

Let's call these type A states $s^a$

And type B states get sampled less, $s^b$

So, in PG, expectation of $s^a$ is accurate, while for $s^b$, it gets overfit!

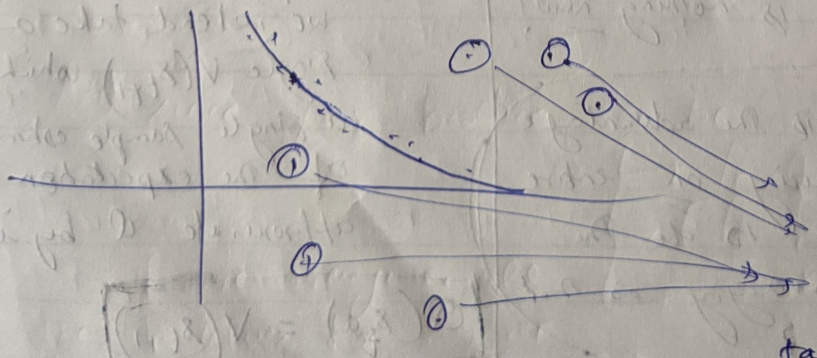
So if a network you get generalization!

all the noise are fitted! infact the noise is taken as the exact value!

overfitting to noise would change the network so badly, that it will effect a lot for $s^a$ the majority so loss will be more!

So, instead, noise is ignored,

$s^a$ are properly fitted, & $s^b$ values follow from the GENERALIZATION



All the noise are IGNORED, Instead fitted values are taken from the generalization!

## Bootstrapped Estimates

We can do TD-learning, Temporal difference learning!

So,

~~$V(s_t) = \mathbb{E}_{s_t} V(s_t)$~~

$$V(s_t) = \mathbb{E}_{a_t} Q(s_t, a_t)$$

$a_t$ → take a sample

$$\approx Q(s_t, a_t)$$ → from the sampled trajectory

$$\approx r_t + \mathbb{E}_{s_{t+1}} V(s_{t+1})$$

$s_{t+1}$ → take a sample

$$\approx r_t + V(s_{t+1})$$ → from the sampled trajectory!

So, take $\boxed{r_t + V(s_{t+1})}$ as the target for $V(s_t)$

**Funny thing is** this is from the network's output!

So, this is the V value for the "previous" policy!

Since in one gradient step, the policies don't change that much!

we can approximate previous policy's V as current policy's V!

So under bootstrapped estimates, if you observe the critic training!

It is

$$V(s_t) \leftarrow r_t + V(s_{t+1})$$

~~So each time the network is expected to predict a high~~

Suppose all rewards are positive in a given problem, no negative rewards!

Then, each time the network is expected to predict a value bigger than its previous prediction!

↳ This would result in explosion, where soon the V function would be outputting $\infty$

So, we need to introduce DISCOUNTING in the rewards of the problem

$$V(s_t) \leftarrow r_t + \gamma V(s_{t+1})$$

---

Now, Baseline

if your eqn uses Q values,

$$\sum_{t=1}^{T} \sum_{a,s \sim \pi_\theta} \nabla_\theta \log \pi_\theta(a|s) \, Q(a,s)$$

And since your baseline can be

- fn of time
- fn of state
- constant

(proved in advanced & basic baselines)

if we average all the Q values for a given state, we get the V value

So, $\sum_{t=1}^{T} \mathbb{E}_{a,s \sim \pi_\theta} \nabla_\theta \log \pi_\theta(a|s) \left[ Q(s,a) - V(s) \right]$

And this is called the **ADVANTAGE**

And it's evaluated or approximated as

$$r(s,a) + \gamma V(s') - V(s)$$

↑ This is next state in the sample

Now, You have options, you can use the critic to evaluate Advantage

OR in "Policy Gradients", you can use critic to evaluate (only) the baseline $V(s)$

OR You can take the $n^{th}$ advantage!

$$V(s_t) = r_t + \gamma r_{t+1} + \dots \gamma^{n-1} r_{t+n-1} + \gamma^n V(s_{t+n})$$

↑ where for n steps, you can use values from the sample & @ the $n^{th}$ step decided to use the critic

OR **Generalized Advantage Estimator**, where you take all n-step advantages ($n=1$ to $T$) & discount ($\lambda$) them!