Apr 22
12:14pm

RL objective

$$\theta^* = \arg\max_\theta \; E_{\tau \sim \pi_\theta(\tau)} [r(\tau)]$$

θ^* = optimal policy's parameters

So, this quantity we need to increase

Let's call the obj, "J"

$$J(\theta) = E_{\tau \sim \pi_\theta(\tau)} [r(\tau)]$$

So this is our Total Reward

⇒ To maximize it, we need to find

$$\frac{\partial}{\partial \theta}(J(\theta))$$

$$\theta = \theta + \frac{\partial}{\partial \theta}[J(\theta)]$$

↑ Gradient Ascent

To evaluate, $J(\theta)$

we need to run samples to collect trajectories

So, $J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} r(\tau_i)$

as $N \to \infty$, $J(\theta)$ becomes equal to this quantity!

So more the samples, more accurate it is!!

dividing by $\frac{1}{N}$, will give us the probability of trajectory $P(\tau_i)$

(Expected r_t)

E_{s_t} P_{a_t} $r(s_t, a_t)$

$J(\theta)$

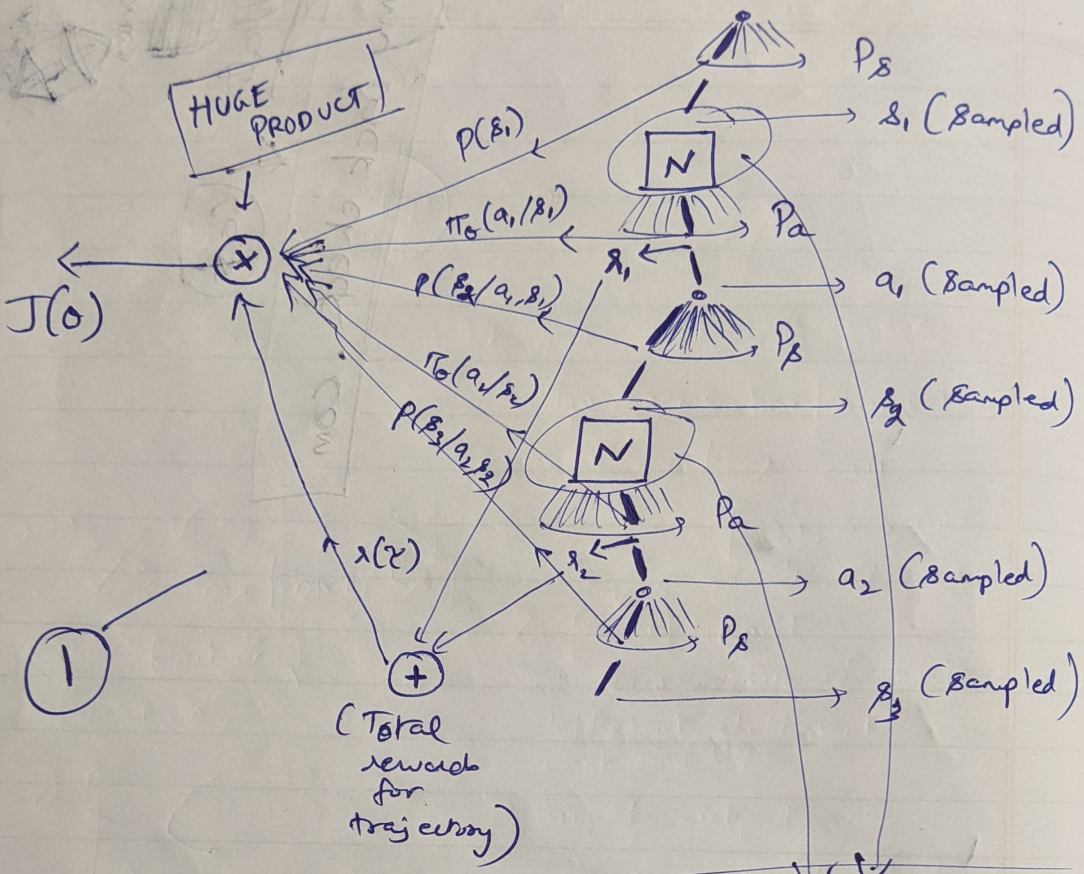
Now,

for

if equa

But

In order to calculate the gradient, we can visualize the computation graph & derive it from there!



But, That's NOT "true"!

Now, if you see, the network is repeating like RNN & the weights are shared!

And, Just like RNN, it looks like 1st network's choices affect future network's choices!

So, multiple gradients will flow for one network's choices from all of future + present!

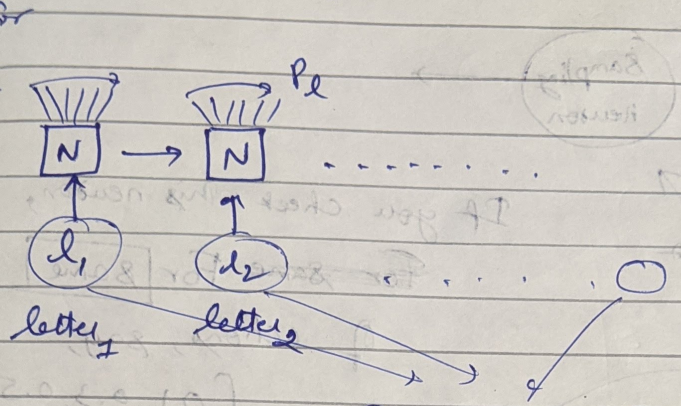
According to this assumption, it would also look like you would need to know the values of environment dynamics like $p(s_2/s_1, a_1)$, $p(s_3/s_2, a_2)$ in order to correctly backpropogate since they look like they are PART of the network!

But, That's also NOT "true"!

Let's take an example to understand why!

In, lec 10, of CS231n, Winter 2016,

Andrej demonstrates a RNN that predicts next letter given ~~current~~ current sequence!



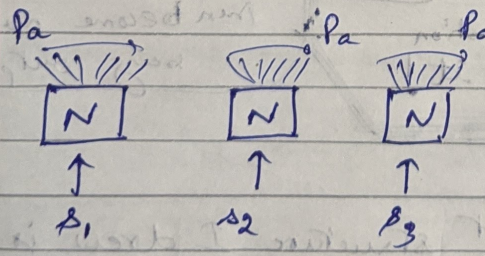
If you look @ all the l_i, These are actually being outputted by the world dynamics! we speak The world dynamics of english language generates these letters!

And what we see as input here, is actually sampled from this world distribution probability!

So, Here too, we don't know the world dynamics

But, we don't even include it as part of our calculation!

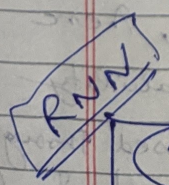
So, in our network, if you incorporate the same, taking all sampling to be INPUT, both action & state!



Then the network BREAKS

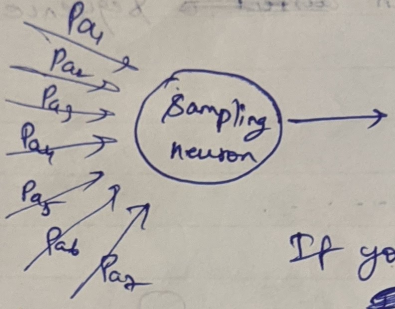
So, 1st network's choices gets NO feedback or gradient from future, although it plays a part in that!

The same would have happened, if ~~Bump~~ Andrej's problem!



But, they cleverly introduced connections b/w the networks, so that the gradient flows

By the way, before we proceed, what is this Sampling step, How do we backpropagate through it?



If you check this neuron,

~~For some~~ For same distribution of actions, say,

$[0.1, 0.3, 0.5, 0.1]$
a_1 a_2 a_3 a_4

So, same input the neuron would output, different different results each time!

it would output

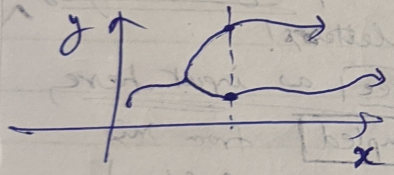
$a_1, a_3, a_2, a_3 \ldots$

different labels each time for same input (distribution)

To make it a believable neuron, you can encode the labels in one-hot representation & it would then become numbers being outputted

So this BREAKS

The definition of a function,



For same x, we can't have 2 values of y!

So Sampling is not even a function, & there can be NO backpropogation Through it!

So, what ch... EXPECTATIO...

And this is what we are optimizing!

So, the 1st structure I drew is not even VALID, Also, it's not even a computation graph because in the step of Sampling, there is NO computation involved, the probability distribution values is not taken & passed through a function where multiplication (or) division happens! In fact, There is NO function that can represent Sampling!

So $\pi(x)$ is INPUT

So, what I drew (is) a sequence graph, NOT computation graph

It's the order in which the events happened!

It's not a flow of computations!

So, All sampling are INPUT!

BOR, { states + actions

if states + actions are input (fixed)

Then even, the rewards (functions of states + actions) are FIXED

So, all 3, (S, a, R(S, a)) are fixed, constants + inputs!

So, for a Trajectory (τ), if states, actions + rewards don't change!

Then by changing the policy! (θ)

The ~~new~~ total reward would still remain the same right! since rewards don't change

So, what changes is EXPECTATION of rewards (not) rewards, !

And this is what we are optimizing!

So, for 2 policies, θ_1 + θ_2

For the same Trajectory (τ)

& the rewards are same! R(τ)

But the expectations of rewards are different!

So R(τ) is INPUT

And we pick the policy whose expectation is HIGHEST

θ, $J(\theta) = \underset{\tau \sim \pi_\theta(\tau)}{E}[r(\tau)]$

discrete

$\sum_{\text{all } \tau} r(\tau) p_\theta(\tau)$ $\qquad$ $\int_{\text{all } \tau,\ -\infty \text{ to } \infty} p_\theta(\tau) r(\tau)\, d\tau$

θ,

$J(\theta) = \sum_{\text{all } \tau} p_\theta(\tau) r(\tau)$

$\frac{\partial}{\partial \theta}[J(\theta)] = \frac{\partial}{\partial \theta}\left[\sum p_\theta(\tau) r(\tau)\right]$

$= \sum_{\text{all } \tau} \frac{\partial}{\partial \theta}\left[p_\theta(\tau) r(\tau)\right]$ ($r(\tau)$: constant)

$= \sum_{\text{all } \tau} r(\tau) \frac{\partial}{\partial \theta} p_\theta(\tau)$

You can think like, when, you keep increasing number of τ by ~~[illegible]~~ increasing your precision limit!

There will be many $r(\tau)$ & $p(\tau)$ approximately SAME for many τs,

θ, when you group them, by taking common, & (the many's count is $d\tau$)

Then, $\sum_{\text{(increasing } \tau \text{ by } d\tau)} p_\theta(\tau) r(\tau)\, d\tau = \int p_\theta(\tau) d\tau$ ($r(\tau)$)

$p(s_1) \times \pi_\theta(a_1/s_1)\, p(s_2/s_1, a_1)\, \pi_\theta(a_2/s_2) \ldots$

environment dynamics are NOT function of θ, s_0, constants!

$= \sum_{\text{all } \tau} r(\tau) \left[p(s_1)\, p(s_2/a_1 s_1)\, p(s_3/a_2 s_2) \ldots\right] \frac{\partial}{\partial \theta}\left[\pi_\theta(a_1/s_1) \times \pi_\theta(a_2/s_2) \ldots\right]$

derivative of products

$\frac{d}{dx}[y_1 \times y_2 \times y_3 \ldots y_n] = y_1' y_2 y_3 \ldots y_n + y_1 y_2' y_3 \ldots y_n + \ldots$

$y_1 y_2 \ldots y_n'$

COMMONS can be computed ONCE

You will have 'n' terms, where in each term, one of the terms is differentiated!

$$\frac{\partial}{\partial \theta_0} = \sum_{\text{all } \tau} R(\tau) \left[p(s_1) p(s_2|a_1 s_1) \cdots \right] \frac{\partial}{\partial \theta_0} \left[\pi_\theta(a_1|s_1)\, \pi_\theta(a_2|s_2) \cdots \right]$$

$$= \left\{ \begin{array}{l} \pi_\theta'(a_1|s_1)\, \pi_\theta(a_2|s_2) \cdots \pi_\theta(a_n|s_n) \\ + \pi_\theta(a_1|s_1)\, \pi_\theta'(a_2|s_2) \cdots \pi_\theta(a_n|s_n) \\ \vdots \\ \pi_\theta(a_1|s_1)\, \pi_\theta(a_2|s_2) \cdots \pi_\theta'(a_n|s_n) \end{array} \right.$$

To compute this is difficult to code + Not efficient! → many of the multiplications is repeated!

~~$\pi_\theta(a_1|s_1) \cdots$~~ $\cdot\, \pi_\theta(a_i|s_i) \times \pi_\theta(a_{i+1}|s_{i+1})$

Duplication of computation ← This product is calculated 'n-2' times!

If ONLY we can compute this SHARED computation ONLY ONCE

$$= \left\{ \begin{array}{l} \dfrac{\pi_\theta'(a_1|s_1)}{\pi_\theta(a_1|s_1)} \times \left[\pi_\theta(a_1|s_1)\, \pi_\theta(a_2|s_2) \cdots \pi_\theta(a_n|s_n) \right] \\ + \dfrac{\pi_\theta'(a_2|s_2)}{\pi_\theta(a_2|s_2)} \times \left[\pi_\theta(a_1|s_1)\, \pi_\theta(a_2|s_2) \cdots \pi_\theta(a_n|s_n) \right] \\ \vdots \\ + \dfrac{\pi_\theta'(a_n|s_n)}{\pi_\theta(a_n|s_n)} \times \left[\pi_\theta(a_1|s_1)\, \pi_\theta(a_2|s_2) \cdots \pi_\theta(a_n|s_n) \right] \end{array} \right.$$

COMMON can be computed ONCE

Apr 22
5:08 pm

So,

$$= \sum_{\text{all } \tau} R(\tau) \left[p(s_1) p(s_2|s_1, a_1) \dots \right] \left[\pi_\theta(a_1|s_1) \pi_\theta(a_2|s_2) \dots \pi_\theta(a_n|s_n) \right]$$

$$\times \left[\frac{\pi_\theta'(a_1|s_1)}{\pi_\theta(a_1|s_1)} * \frac{\pi_\theta'(a_2|s_2)}{\pi_\theta(a_2|s_2)} \dots \frac{\pi_\theta'(a_n|s_n)}{\pi_\theta(a_n|s_n)} \right]$$

$= p_\theta(\tau)$; "integral"

$$\nabla_\theta = \sum_{\text{all } \tau} R(\tau) p_\theta(\tau) \left[\frac{\partial}{\partial \theta} \left[\log \pi_\theta(a_1|s_1) \right] \dots \right]$$

$$\nabla_\theta \left[\underset{\tau \sim \pi_\theta(\tau)}{E} \left[\sum_{i=1}^{T} \frac{\partial}{\partial \theta} \left(\log \pi_\theta(a_i|s_i) \right) \right] R(\tau) \right]$$

$$\frac{\partial J(\theta)}{\partial \theta}$$

Policy Gradients

Now, if **partial observability** is included

Then,

$$= \sum_{\text{all } \tau} p_\theta(\tau) R(\tau)$$

$$p(s_1) p(o_1|s_1) \pi_\theta(a_1|o_1) p(s_2|s_1, a_1) \; p(o_2|s_2) \pi_\theta(a_2|o_2) \dots$$

Extra 'o' dynamics are added! → These are also CONSTANTS!

[observations]

So, it comes out of differentiation!

And formula doesn't CHANGE