Policy Gradient

Goal of reinforcement learning and how to maximize the reward function

- agent communicate with environment through reinforcement leanring: when agent is in state s, it will get corresponding action a according to a policy $\pi_\theta(a|s)$ which is parametrized by θ. Then it can get new state based on transition function $p(s'|s,a)$, then loop over the operations.

- According to the property of Markov (the state of s + 1 is only related to the state of state s). The probability of a specific trajectory is $p_\theta(\tau)=p(\mathbf{s}*1)\prod*{t=1}^T\pi_\theta(\mathbf{a}_t|\mathbf{s}*t)p(\mathbf{s}*{t+1}|\mathbf{s}_t,\mathbf{a}_t)$. (The first state is given).Then here comes the goal of reinforcement learning : Find the optimal parameter θ^* to maximize the reward function wrt trajectory:

$$ \max_\theta\mathbf{E}*{\tau\sim p*\theta(\tau)}\left[\sum_tr(\mathbf{s}_t,\mathbf{a}_t)\right] $$

First consider the objection function as $$ J(\theta)=\mathbf{E}*{\tau\sim p*\theta(\tau)}\left[\sum_tr(\mathbf{s}_t,\mathbf{a}_t)\right] $$

- Unless we our distribution of the trajectory is very clear(eg: Gaussian distribution), we cannot precisely evaluate the expection, but we can use Monte Carlo to approximate. The easiest way, we sample from the trajectory, an unbiased estimation will be $$ \frac{1}{N}\sum_i\sum_tr(\mathbf{s}*{i,t},\mathbf{a}*{i,t}) $$
- N is the quantity of sample, $s_{i,t}$ is the state of i^{th} sample, time t. so as the action$a_{i,t}$. We can get each sample(trajectory) $\sum_tr(\mathbf{s}*{i,t},\mathbf{a}*{i,t})$, then take the expection of the score.

How to optimize the $J(\theta)$? gradient!

- The intuition of the gradient is to take the first gradient wrt parameter, and try to target this step in objective function. Define reward is $r(\tau)=\sum_tr(\mathbf{s}*t,\mathbf{a}*t)$, then $J(\theta)=\mathbf{E}*{\tau\sim p*\theta(\tau)}[r(\tau)]$. It can also be written as $$ J(\theta)=\int p_\theta(\tau)r(\tau)\mathrm{d}\tau $$
- Take the gradient wrt to θ. $\nabla_\theta J(\theta)=\int \nabla_\theta p_\theta(\tau)r(\tau)\mathrm{d}\tau$. For $\nabla_\theta p_\theta(\tau)$, we can find that:

$$ p_\theta(\tau)\nabla_\theta \log p_\theta(\tau)=p_\theta(\tau)\frac{\nabla_\theta p_\theta(\tau)}{p_\theta(\tau)}=\nabla_\theta p_\theta(\tau) $$

- then plug in this formular, we can get $$ \nabla_\theta J(\theta)=\int p_\theta(\tau)\nabla_\theta \log p_\theta(\tau)r(\tau)\mathrm{d}\tau $$
- why do we make this changes? since we have the $p_\theta(\tau)$, the function of taking gradient of $J(\theta)$ can be turning back to : $$ \nabla_\theta J(\theta)=\mathbf{E}*{\tau\sim p*\theta(\tau)}[\nabla_\theta \log p_\theta(\tau)r(\tau)] $$
- Go back to the objective function, what we what to do is to get the parameter θ, then we can maximize the reward function. The way we want to get the parameter has the similarity with maximum likelihood. We know that the probability density function is $p_\theta(\tau)=p(\mathbf{s}*1)\prod*{t=1}^T\pi_\theta(\mathbf{a}_t|\mathbf{s}*t)p(\mathbf{s}*

{t+1}|\mathbf{s}t,\mathbf{a}t)$. Take the Log on both side, then we get: $$ \log p\theta(\tau)=\log p(\mathbf{s}1)+\sum{t=1}^T[\log \pi\theta(\mathbf{a}_t|\mathbf{s}t)+\log p(\mathbf{s}{t+1}|\mathbf{s}_t,\mathbf{a}_t)] $$

- From this result, we can see that the initial state $\log p(\mathbf{s}1)$ and transitional probability $\log p(\mathbf{s}{t+1}|\mathbf{s}t,\mathbf{a}t)$ has nothing to do with θ. Then we can siplified it as: $$ \nabla\theta\log p\theta(\tau)=\sum_{t=1}^T\nabla_\theta\log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) $$
- Then finally we can plug in back to the gradient of $J(\theta)$ (wrt θ).

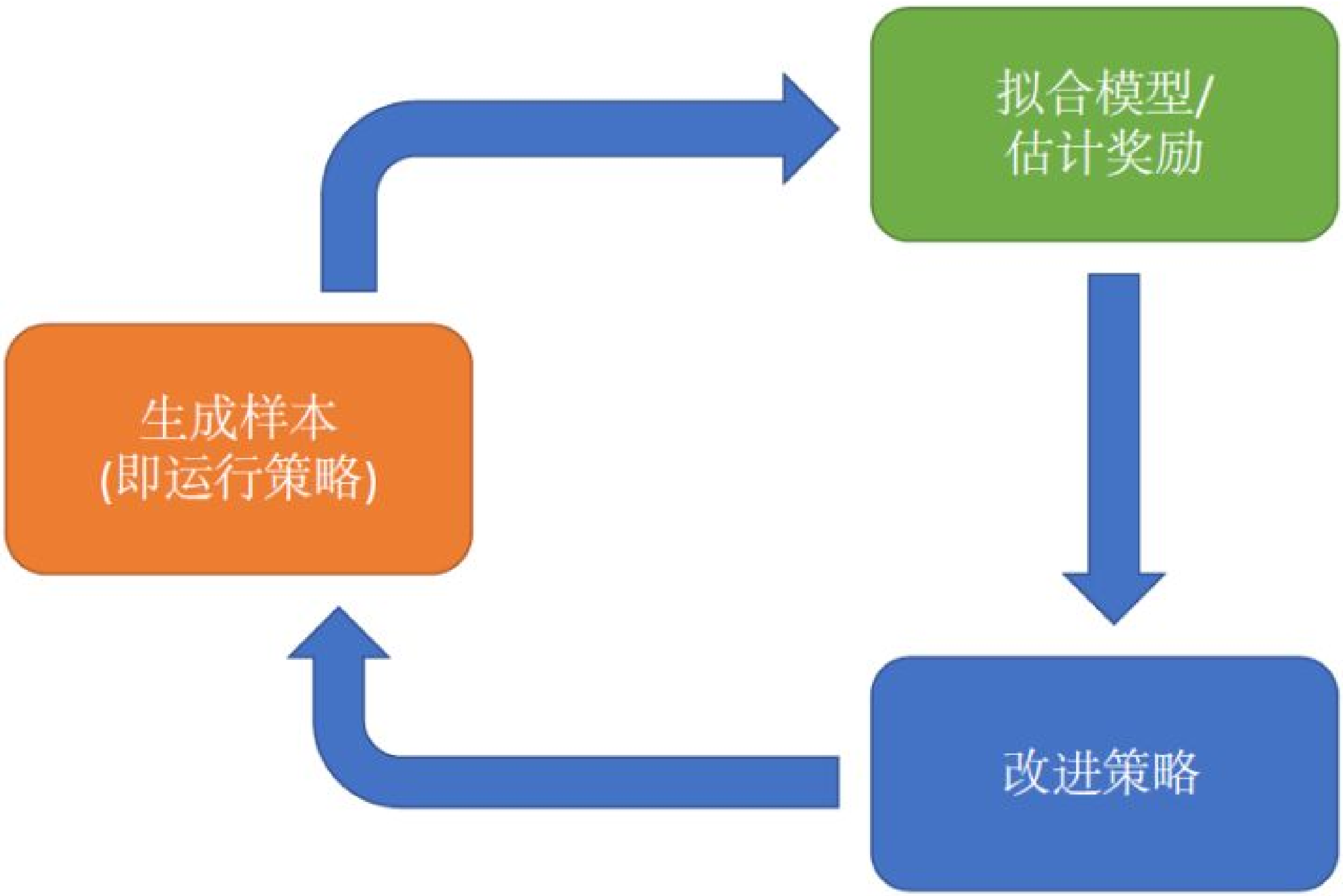
$$ \nabla_\theta J(\theta)=\mathbf{E}{\tau\sim p\theta(\tau)}\left[\left(\sum_{t=1}^T\nabla_\theta\log \pi_\theta(\mathbf{a}_t|\mathbf{s}t)\right)\left(\sum{t=1}^Tr(\mathbf{s}_t,\mathbf{a}_t)\right)\right] $$

- The good property of these formula is that we don't need to know initial distribution and transition function anymore.

- Then using Monto Carlo we get an unbiased estimation: $$ \nabla_\theta J(\theta)\approx\frac{1}{N}\sum_{i=1}^N\left[\left(\sum_{t=1}^T\nabla_\theta\log \pi_\theta(\mathbf{a}{i,t}|\mathbf{s}{i,t})\right)\left(\sum_{t=1}^Tr(\mathbf{s}{i,t},\mathbf{a}{i,t})\right)\right] $$

- After getting θ, we can use graident ascent to optimize.

- From the image below, we can see that a classical reinforcement learning process



1. for step one, running the policy $\pi_\theta(\mathbf{a}|\mathbf{s})$, get the sample ${\tau^i}$
2. Then estimate the gradient of $$ \nabla_\theta J(\theta)\approx\frac{1}{N}\sum_{i=1}^N\left[\left(\sum_{t=1}^T\nabla_\theta\log \pi_\theta(\mathbf{a}{i,t}|\mathbf{s}{i,t})\right)\left(\sum_{t=1}^Tr(\mathbf{s}{i,t},\mathbf{a}{i,t})\right)\right] $$

3. Then gradient ascent: $\theta\leftarrow\theta+\alpha\nabla_\theta J(\theta)$

for the output, if your data is discrete, then it could be softmax, or it's continuous, output the parameters distribution

1. How to optimize the $J(\theta)$? gradient!
2. Has the similarity with maximum likelihood
3. MDP actually not used
4. Reduce the number and it can lower your variance
5. Baselines: Take the good stuffs, and make it more likely, and take the bad stuffs and make it less likely. Average reward is not the best baseline, but it's prettu good
6. Take the integral of a normalize distribution, and the result is 1
7. Analyzing variance: THe best baseline is just expected reward, but weighted by graident magnitudes.
8. Policy gradient is on-policy: each time you genrate new policy, and generate new sample, and you discard your old samples
9.