Policy gradient methods.

In this document, an introduction to policy gradient methods is given. This introduction is based on following posts.

1. http://machinelearningmechanic.com/deep_learning/reinforcement_learning/2019/12/06/a_mathematical_introduction_to_policy_gradient.html
2. https://danieltakeshi.github.io/2017/03/28/going-deeper-into-reinforcement-learning-fundamentals-of-policy-gradients/
3. https://www.janisklaise.com/post/rl-policy-gradients/

Derivation of the vanilla policy gradient method

As we discuss only model free RL methods, we don't know what are the transition probabilities $P(s_{t+1} \mid s_t, a_t)$ of the Markov Process (MP). The idea of policy gradient methods is to directly interact with the environment (on-policy) and utilise the rewards obtained as well as the observed transitions to find an optimal policy.

In order to find the optimal policy, we can craft an objective function that is a function of reward as follows,

$$J(\theta) \approx \frac{1}{L} \sum_{\tau} \sum_{t=0}^{T-1} \gamma^t r(s_{\tau,t}, a_{\tau,t}) = \mathbb{E}_{\pi_\theta}(\sum_{t=0}^{T-1} \gamma^t r_t) - (1)$$

Where $r(s_{\tau,t}, a_{\tau,t})$ is the reward for a given trajectory $\tau = (s_0, a_0, s_1, a_1, \ldots, s_{T-1}, a_{T-1}, s_T)$. The T in the trajectory indicated the terminal time step and s_T is the terminal state. The last action is a_{T-1}. As the objective, we find the mean over L number of trajectories and all the L trajectories are followed using the same policy π_θ. The π_θ is the probability of taking an action given the a state determined by the policy parameterised with θ (policy network parameters)

So now our goal is to find the model parameters θ that maximise the reward expectation $max_\theta(\mathbb{E}_{\pi_\theta}[\sum_{t=0}^{T-1} \gamma^t r_t])$.

Let's take $R(\tau) = \sum_{t=0}^{T-1} \gamma^t r_t(\tau)$ the total discounted reward (total return) for a given trajectory τ.
Now we can write (1) as $J(\theta) = \mathbb{E}_{\pi_\theta} R(\tau)$. So now our goal is to increase this expectation, we can use gradient ascent. Therefore let's find the gradient of the objective function as follows.

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\pi_\theta} R(\tau)$$

Now remember, the environment is stochastic where the agent will transit from one state to another with a given probability, therefore, for a given policy π_θ, there could be different τ trajectories.

$\nabla_\theta \mathbb{E}_{\pi_\theta} R(\tau) = \nabla_\theta \sum_{\tau} P(\tau \mid \theta) R(\tau)$ where the $P(\tau \mid \theta)$ is the probability of a trajectory given the model parameters. Since the gradient of a sum = sum of the individual gradients,

$$\nabla_\theta \mathbb{E}_{\pi_\theta} R(\tau) = \sum_{\tau} \frac{P(\tau \mid \theta)}{P(\tau \mid \theta)} \nabla_\theta P(\tau \mid \theta) R(\tau) = \sum_{\tau} P(\tau \mid \theta) \frac{\nabla_\theta P(\tau \mid \theta)}{P(\tau \mid \theta)} R(\tau)$$

Now from the log derivative trick,

$$\nabla_\theta \mathbb{E}_{\pi_\theta} R(\tau) = \sum_\tau P(\tau \mid \theta) \nabla_\theta log P(\tau \mid \theta) R(\tau)$$

$$\nabla_\theta \mathbb{E}_{\pi_\theta} R(\tau) = \mathbb{E}_{\pi_\theta} \nabla_\theta log P(\tau \mid \theta) R(\tau) - (2)$$

The probability of a trajectory can be expressed as a chain of probabilities that satisfies the Markov property. The trajectory probability can be calculated as follows,

$$P(\tau \mid \theta) = \mu(s_0) \prod_{t=0}^{T-1} P(s_{t+1} \mid s_t, a_t) \pi_\theta(a_t \mid s_t)$$

The μ represents the starting state distribution and does not depend on any action. The rest of the states depends on the action and state transition probabilities. If we take a deterministic action $\pi(a_t \mid s_t) = 1$ and deterministic transitions $P(s_{t+1} \mid s_t, a_t) = 1$ the existence of the trajectory τ, $P(\tau \mid \theta) = 1$. Likewise once we have stochastic policies and state transition the existence of the trajectory τ would vary according to the policy and environment dynamics.

Now when we take the derivative of the trajectory probability as follows,

$$\nabla_\theta \log P(\tau \mid \theta) = \nabla_\theta \log(\mu(s_0) \prod_{t=0}^{T-1} P(s_{t+1} \mid s_t, a_t) \pi_\theta(a_t \mid s_t))$$

$$\nabla_\theta \log P(\tau \mid \theta) = \nabla_\theta [\log \mu(s_0) + log \prod_{t=0}^{T-1} P(s_{t+1} \mid s_t, a_t) \pi_\theta(a_t \mid s_t)]$$

$$\nabla_\theta \log P(\tau \mid \theta) = \nabla_\theta [\log \mu(s_0) + \sum_{t=0}^{T-1} (log P(s_{t+1} \mid s_t, a_t) + log \pi_\theta(a_t \mid s_t))]$$

Now moving the gradient inside:

$$\nabla_\theta \log P(\tau \mid \theta) = \nabla_\theta \log \mu(s_0) + \sum_{t=0}^{T-1} (\nabla_\theta log P(s_{t+1} \mid s_t, a_t) + \nabla_\theta log \pi_\theta(a_t \mid s_t))$$

As one can see the μ and P does not depend on the θ parameters, hence their gradients $\nabla_\theta = 0$. Then all we left with is the policy part

$$\nabla_\theta \log P(\tau \mid \theta) = \nabla_\theta \sum_{t=0}^{T-1} log \pi_\theta(a_t \mid s_t) - (3)$$

The model free property comes since the model does not depends on the environment (because for the functions of the environment the gradients are $\nabla_\theta = 0$)

Now combining the equations (2) and (3) we get the following equation,

$$\nabla_\theta \mathbb{E}_{\pi_\theta} R(\tau) = \mathbb{E}_{\pi_\theta} [R(\tau) \nabla_\theta \sum_{t=0}^{T-1} log \pi_\theta(a_t \mid s_t)]$$

By expanding the expectation we can get the objective function we want to increase,

$$\nabla_\theta J(\theta) \approx \frac{1}{L}\sum_{\tau}\sum_{t=0}^{T-1}\nabla_\theta log\pi_\theta(a_t|s_t)R(\tau) - (4)$$

Now once we know the policy gradients from equation (4) we can use the gradient ascend rule to update the parameters

$\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$ where α is the learning rate $-$ (5)

By looking at the equation (4) one can notice that the each gradient of the log policy is multiplied by the total reward obtain for that trajectory. Which means all the good or bad action are weighted by the total reward obtain in the episode. Now think of a case where the reward is given at the end of an episode as 1 or -1, in this case its ok to weigh all the steps of the policy with 1 or -1 and guide the parameters θ to find the optimal policy through gradient ascent. But if we get a reward for each step we take in the episode, ultimately we end up with an accumulated reward which could be large (like 10 or -10). This large reward won't change the direction of the gradients but rather the magnitude. Now if this magnitude is very large, our update in (5) will be very large as well. This is not desirable since it introduces a high variance to the gradients.

Another issue of this reward accumulation is, when we multiply the log policy at a step with the total reward, we are considering that the past actions also depends on the reward obtain in the future, which is not the case (causality).

The reward accumulation for a given trajectory τ_0 at each step can be visualised using the following backup diagram,

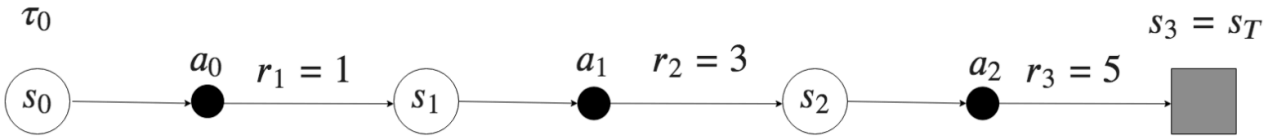

Fig-1 (Ref.1) Backup diagram of a trajectory τ_0

$R_{0,0} = R_{0,1} = R_{0,2} = r_1 + r_2 + r_3 = 9$; here the $\gamma = 1$

Variance reduction methods

In order to reduce the variance in gradients, there are two main methods introduced. They are the reward to go (to address the causality in the action reward system) and the advantage method (which estimates a state dependent baseline or take the mean reward as the baseline)

The reward to go method for a given trajectory can be visualised as follows,

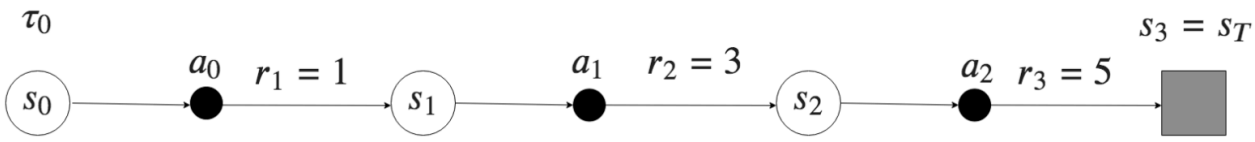

Fig-1 (Ref.1) Backup diagram of a trajectory τ_0

The reward-to-go at each time step is (here the $\gamma = 1$):

$R_{0,0} = r_1 + r_2 + r_3 = 1 + 3 + 5 = 9$

$R_{0,1} = r_2 + r_3 = 3 + 5 = 8$

$R_{0,2} = r_3 = 5$

In the above examples the $\gamma = 1$ for simplicity.
In contrast to the vanilla policy gradient, the reward to go method considers the reward accumulation with respect to the causality property.

Now considering this time dependent property of reward accumulation, we can alter the equation (4) and obtain,

$$\nabla_\theta J(\theta) \approx \frac{1}{L} \sum_{\tau} \sum_{t=0}^{T-1} \nabla_\theta log \pi_\theta(a_t | s_t) \sum_{t'=t}^{T-1} \gamma^{t'} r_{t'}(\tau) \quad -(6)$$

Baselines

However, we can still get large gradients even with the reward-to-go method. So in order to further reduce the variance, the advantage method can be used. In the advantage method, one can either use a state independent or state depended value to estimate the advantage of an action.

In the state independent case, we simply subtract the mean reward for an episode from the reward at each step. The variance reduction in this case happens as follows,

To see the intuition behind the state independent method, let's take two action a_1, a_2 where at each action, the accumulated reward (now on will call this the return) is 100 and 102. The mean reward for these two actions is 100. So now if we subtract the baseline from each actions, we yield a return of -1 and 1 for action a_1 and a_2. So now the gradient ascent method know that taking the actions that moves the gradients in the positive direction is better than the actions that would move a gradient in the negative direction. With this baseline reduction method, the equation (6) becomes,

$$\nabla_\theta J(\theta) \approx \frac{1}{L} \sum_{\tau} \sum_{t=0}^{T-1} \nabla_\theta log \pi_\theta(a_t | s_t)(\sum_{t'=t}^{T-1} \gamma^{t'} r_{t'}(\tau) - b) \quad -(7)$$

Where the advantage is defined as,

$A(s_t, a_t) = \sum_{t'=t}^{T-1} \gamma^{t'} r_{t'}(\tau) - b$, b is the mean reward of the episode in this case

Actor Critic Method

Now if we look at a state space of an MP, one would realise that it is better to be in some states than the others (because these states would give a good reward). Taking this concept, we can measure a state dependent advantage for a given action. Intuitively the better the advantage of being in a particular state, the action that leads to that state should be more optimal.

This is achieved by calculating the advantage of an action at each state. To calculate the advantage, we can use another network to approximate the expected state value. The advantage function is as follows,

$$A(s_t, a_t) = \sum_{t'=t}^{T-1} \gamma^{t'} r(s_{t'}, a_{t'}) - \hat{V}_\phi(s_t) \quad (8)$$

The $\hat{V}_\phi(s_t)$ is an estimation of the state value. The parameters of the state estimator can be learnt using the mean square error where the targets would be $\sum_{t'=t}^{T-1} \gamma^{t'} r(s_{t'}, a_{t'})$ and the prediction would be $\hat{V}_\phi(s_t)$.

This is the main idea behind the actor critic method. The actor network predicts a policy at each training step and the critic network predicts an expected state value. So now the idea of this advantage function is to know how advantages it is to take a particular action at a state compared to the expected value of the state (expected over all possible actions).

However, the issue with calculating the advantage using equation (8) is we still have to wait till the episode ends (these type of methods are called Monte-Carlo (MC) methods, vanilla policy gradient is one of them). To address this issue, we can bootstrap the target with the state value estimator (like in the DQN case we bootstrap with a target network).

So once the advantage is calculated using bootstrapping (these types of methods are call Temporal Difference (TD) methods) we can update our policy network + the state value estimation network at each step (online learning). The bootstrapped advantage function is as follows,

$$A(s_t, a_t) = r(s_t, a_t) + \gamma \hat{V}_\phi(s_{t'}) - \hat{V}_\phi(s_t) \quad (9)$$

Note that it is also possible to bootstrap the equation using a target network for stability like in the DQN case. As one can see, when the action at the time step t is known, $r(s_t, a_t) + \gamma \hat{V}_\phi(s_t)$ is the action value $q(s_t, a_t)$ at state s_t. So now we can rewrite the equation (9) as,

$$A(s_t, a_t) = Q(s_t, a_t) - \hat{V}_\phi(s_t) \quad (10)$$

Where from (10), the advantage is the difference between the action value and state value

Now similar to the vanilla policy gradient with reward-to-go with baseline, we can get the new objective function with the advantage as follows,

$$\nabla_\theta J(\theta) \approx \frac{1}{L} \sum_\tau \sum_{t=0}^{T-1} \nabla_\theta log \pi_\theta(a_t | s_t)[Q(s_t, a_t) - V_\phi(s_t)]$$

$$\nabla_\theta J(\theta) \approx \frac{1}{L} \sum_\tau \sum_{t=0}^{T-1} \nabla_\theta log \pi_\theta(a_t | s_t) A(s_t, a_t)$$

Reference:

Ref.1: http://machinelearningmechanic.com/deep_learning/reinforcement_learning/2019/12/06/a_mathematical_introduction_to_policy_gradient.html

Ref.2: https://danieltakeshi.github.io/2018/06/28/a2c-a3c/

Ref.3: https://julien-vitay.net/deeprl/ActorCritic.html

Ref.4: https://www.freecodecamp.org/news/an-intro-to-advantage-actor-critic-methods-lets-play-sonic-the-hedgehog-86d6240171d/

Ref.5: