Policy-Based Reinforcement Learning Approaches: Stochastic Policy Gradient and the REINFORCE Algorithm

Chapter · June 2019

DOI: 10.1007/978-981-13-8285-7_10

CITATIONS
16

READS
1,404

1 author:

Mohit Sewak
Microsoft
94 PUBLICATIONS **874** CITATIONS

SEE PROFILE

Citations

BibTex:

```
@incollection{drl-10-policy-based-approaches,
  title={Policy-Based Reinforcement Learning Approaches},
  author={Sewak, Mohit},
  booktitle={Deep Reinforcement Learning},
  pages={127--140},
  year={2019},
  publisher={Springer}
}
```

Plain Text:

M. Sewak, “Policy-Based Reinforcement Learning Approaches”, Deep Reinforcement Learning, pp. 127–140, Springer, 2019.

2

Policy-Based Reinforcement Learning Approaches

Stochastic Policy Gradient and the REINFORCE algorithm

Abstract

In this chapter we will cover the basics of the policy-based approaches especially the policy-gradient based approaches. We will understand why policy-based approaches are superior to that of value-based approaches under some circumstances and why they are also tough to implement. We will subsequently cover some simplifications that will help make policy-based approaches practical to implement and also cover the REINFORCE algorithm.

Introduction to policy-based approaches and Policy Approximation

Until now in this book we focused on estimating different types of Value. Initially we focused on state-value computation (as in Classical-DP), and then state-value estimation (as in TD-Learning). Subsequently we drew our focus towards action-value estimation (as in SARSA and Q-Learning), and finally advantage (incremental value) estimation (as in Dueling DQN).

While discussing Q Learning, we also introduced the concept of value-approximation. Since the value function in Q Learning and similar algorithms are modeled using function-approximators (or machine learning models, as we otherwise refer to) therefore we called it value estimation process instead of value computation or determination process. Any such value estimation process, which is modeled using a value-estimator and is not exact introduces some bias. Also note that we modeled the value as output of approximation ***functions*** (as in Q Learning, DQN, etc.) and started denoting the value (state-value or action-value) as a function of the parameter of these functions. In the case of deep-learning based function-approximators (models) these parameters are the network weights which needs to be optimized.

Maximizing the Reinforcement Learning reward, under such notations would have been equivalent to finding the set of weights which ***'optimizes'*** (as in mathematical optimization) these total rewards received. This is done by minimizing the respective losses as an outcome of the training. Since in most of these occassions the approximation-function was (purposely) differentiable, one of the best suited method to optimize these functions proves to be computing the gradients of the expectancy of reward function (by differentiating them) and then moving in the direction of the gradient till a local 'maximum is achieved so as to maximize the reward expectancy.

The non-classical Reinforcement Learning system is often a combination of solutions to the underlying estimation problem and control problem. Optimization of the estimation model's loss function consummates the training of our function-approximator which could later be deployed to estimate the 'optimal' value of a state or action and serves to provide solution for the estimation sub-problem. The estimations from such value estimation model ***'combined with a specific policy'*** (the policy provides solution to the control sub-problem) helps our agent take an appropriate action. We also discovered the on-policy and off-policy methods, which would either have an inbuilt mechanism to strike a good balance between the exploration of new states/ actions or exploiting the existing training or a different behavioral policy for that determines when to use the estimated values greedily and when to explore further. So essentially the value-estimation/approximation process

finally led to a single 'action' being suggested corresponding to any given state that the agent is in. This action is either directly determined or greedily chosen (as in epsilon-greedy for instance) from the estimation process. So, the 'policy' which directly results in the agent's action, was not our focus throughout. Instead the 'value' which helped form the basis of a *mostly deterministic* 'policy' was the key-focus until now.

Now just take a moment's pause and imagine why we were trying to estimate the value and approximate the ***value-function*** (state-value or action-value function) when the whole intention was to take the best action. Remember we had a similar discussion earlier when we moved from state-value computation/ estimation to action-value computation/ estimation. Until then though we were still focusing on estimating the value but reasoned that since the ultimate goal is to determine the best action, therefore the action-value estimation based-approach could be a more direct means of achieving our goal than following the state-value estimation based-approaches. Using the same reasoning forward with respect to policy, we can argue that approximating the policy function could be a more direct means of achieving our goal than approximating the value function as we have been doing so far. So essentially, we would now like to parametrize the policy itself as below.

\[\pi_{\theta}{(a | s)} = \Pr [a | s; \theta] \] --- (1)

$$\pi_\theta(a|s) = \Pr[a|s;\theta]$$

This is the intuition behind most of the Policy-Approximation based approaches that we will discuss in this book. Policy-Gradient based approaches are very popular under policy-based Reinforcement Learning paradigm. The Policy-Gradient based approaches under Policy-Approximation would mostly try to leverage the property of differentiability (and hence computing gradient) of the (approximate) policy-function to optimize it. Also, as in the case of value-approximation, we would proceed with the focus on model-free (here the term model-free refers to the fact that we may not be in a position to mathematical model the MDP completely and hence have to rely on approximating it instead of knowing it) assumptions for policy-approximations.

Broad difference between value-based and policy-based approaches

The basic difference between the value-based and the policy-based approaches is that in the value-based approaches we learn a 'value-function' from which the 'policy' is derived either explicitly or implicitly. Whereas in the policy-based approaches there is no need to learn or derive the value-function, and we learn the 'policy' directly. The value-function is essentially non-existent in policy-based approaches. Though there exist some variants of hybrid approaches as well, but for now we will use this simple theme for the purpose of drawing intuitions for this chapter.

In 'value-based' approaches we derived the policy on the basis of the estimates generated by an optimal (well trained) value function. Although the value-function was stochastic (that is it generates probabilistic estimate for selecting different actions in a particular state using a given value approximation model assuming no further updates to the model), the so implied policy in most of the implementation of value-based approaches has been mostly deterministic. This is because the policy in the case of value-estimation approaches suggests a single action. This action could be either the best action suggested by the estimation function or any random action, but it lacks the suggested probability distribution for selecting across different actions.

Whereas in the case of 'policy-approximation' approaches, since the 'policy' itself is parametrized, it is 'stochastic' (refer to the equation (1)). This essentially means that for a given state, the policy may have varying probabilities of choosing different actions instead of choosing the 'single' most-optimal action. So, the 'policy-based' approaches would essentially draw samples from this 'stochastic-policy' to refine their estimate of the policy parameter vector θ in a direction (as in the gradient) so as to optimize the policy which would subsequently enable the agent that follows such policy to accumulate maximum cumulative reward. Again, there is a dedicated series of models (like deterministic policy gradient and variants) which are exceptions to this principle. But for now, we will use this distinction to build our intuition further for the purpose of this chapter.

Since the value-based methods have *'deterministic'* or 'non-stochastic' policy, therefore it could select only one action. This has a similar effect of choosing one action with probability 1 and others with probability 0. Even in cases where the value difference between different actions is arbitrary, and hence also negligible or very small, such sort of deterministic (and hence absolute) action choice leads to discontinuous changes in the approximated function, leading to challenges with ***'convergence-assurance'*** of algorithms following the value-function approach.

This shortcoming is not so much pronounced in the case of policy-based approaches as they could stochastically suggest multiple actions with appropriate probabilities.

In case if the importance of this distinction for practical application is not well realized, let us understand this with an example. Suppose during the initiation of a sport (say cricket) match the captain has to 'choose' (takes action of choosing) between 'heads' or 'tails' in a toss (assumingly of an unbiased coin). If he wins the toss then he has to take decision pertaining to that match (say to bat or to ball first). Although the first decision (calling heads or tails) is supposed to be random and arbitrary, and only the second decision (match strategy of batting or balling first) is something that could influence the outcome of the match, using a deterministic policy will end up forcing the agent to learn and determine whether calling 'heads' is better or calling 'tails' is better in a toss given a particular state of the match conditions and team compositions. As the readers would have realized that here state (comprising of the match conditions and team compositions) will have no bearing on the call decision for the toss. By forcing 'deterministic' proposal for actions/ decision we imply that there exists only 'one' optimal action/ decision, and restrict the stochastic recommendations across multiple actions/ decisions choose one. A stochastic policy in this case could have recommended that with 50% probability we call 'heads' and with the remaining 50% we call tails. If we force to train the approximation-function under the assumption of existence of a 'deterministic' policy, where the underlying optimal 'policy' is actually 'stochastic', such approximation-function's training should obviously *not* converge. With reference to our example, a convergence in training would have meant that we successfully trained a model that could clearly identify with high degree of precision and recall that one action is better than all the other actions possible in a particular state as we proceed with the training. Since such observations are contrary to the ground truth, an attempt to train the model under such assumptions should fail to converge.

Another distinction between the value-based approaches and the policy-based approaches is that value-based approaches are suited mainly for scenarios with *small* and *discreet* action-space. In case of ***large action-space or continuous action-space***. A continuous action-space could be considered as a special case of large-action space, with action-space cardinality growing towards infinity. We have already hinted in the earlier chapters some intuition for the reasons behind this observation while discussing the algorithms in the value-based approaches, whereas other reasons could also be implicitly understood from our discussion on stochastic action probability in this section earlier.

Just to understand the reasons related to stochastic action probability which makes the policy-based approaches more suitable for applications with large and continuous action spaces as compared to value-based approaches let us take an example. Proceeding from our initial example from the Cricket theme, let us assume that our agent needs to determine the optimal angle at which the batsmen should

play a stroke on a particular delivery to ensure that the ball reaches the boundary. Each delivery from its start to the result of the result of that delivery is an episode in this example. Assume the state here comprises of two information. The first information is static for a given delivery/ episode and is updated as the baller starts to ball. The other state information is dynamic which updates at a particular frequency, (say once every $1/100^{th}$ second). The static information types are the ball condition (say age, roughness), pitch condition, field placements occupied or vacant (one hot encoded vector). Whereas the dynamic state comprises of the ball velocity, ball direction, the swing/spin of the bowl, and the thrust generated by the swing of the bat, distance between the closest point of bat and ball. The desired decision output from the agent is the angle at which the bat should swing (for the given thrust and timing of the bat) to ensure a boundary. Assuming that the decision needs to be precise to a minute of an angle (1^0 = 60"), the action-space here if discreet discreet could be considered as a vector of dimension 360 x 60 = 21,600 assuming the angle could be between [0^0-360^0]. Alternatively, the action-space here ideally should not be a discreet action space but a continuous one in the range [0^0-360^0]. This is because there is little difference in the outcome between the bat swing between any two contiguous angles (precise up to a minute of an angle), say 45^0.0" and 45^0.1", but a discreet action space, especially in the case of value-based approaches have to select one of these angles and reject the another. From the explanation until now it should be clear why some action criteria are better suited and more practical to be conceived as a continuous action space even when theoretically we could conceive them as a discreet action space with very high cardinality.

Continuing forward with the above example, let us assume that there is a fielder placed in an isolated area of the field where the field boundary is the shortest. Let the fielder's angle from the batting point is 45^0.0". Assuming the players reach to be similar, say $5^0$0", in both the clockwise and anticlockwise directions, we have equally high probability of scoring the boundary at angles < 45^0.0" - $5^0$0" (= 40^0.0") and angles > 45^0.0" + $5^0$0" (= 50^0.0"). In the case of policy-based approaches where policy is stochastic, a similar probability distribution across these decision boundaries could reflect this phenomenon. But in the case of value-based approaches, even if the value of scoring the boundary may reflect a similar stochastic pattern across the discreet action-spaces around these angles, but the deterministic action policy could only select one of the actions and has the reject the other. This distinction is very important for cases where the action suggestions of the agent has to feed into another intelligent system/ agent/ model that works on a large strategical decision-process/ prediction to which this agent's suggestions is one of the many inputs.

As the readers would have realized from the above example why the capability of taking actions in a continuous action-space is so useful, and how the agents using policy-based approaches are naturally suited to provide more effective outcomes under these challenges. But the inherent capability of working well with large/ continuous action-space for the policy-based methods as compared to the value-based

methods comes at a cost. For one, while ‘converging’, the policy-based approaches could ‘converge’ to a local ‘optimum’, instead of a ‘global optima’, and for another ‘evaluating’ a policy has very high variance and could be highly inefficient for the policy-based approaches.

Problems with calculating the Policy Gradient

From the discussion in the previous section it should be evident that the policy-based approaches offer some valuable advantages over the value-based methods and are worth considering. In scenarios where learning the policy directly may be simpler or otherwise more effective and in other scenarios where the action-space is large or continuous the policy-based approaches over distinctive advantages beyond performance/ accuracy over value-based approaches.

Also, as we discussed in the previous section, in policy-based methods, we are interested in optimizing the policy approximator functions in order to maximize the rewards. Also, we learnt that to optimize the policy approximator we may need the gradient of the approximation function that represents this stochastic policy as moving in the direction of this gradient may help minimize the losses and hence optimize the policy. Alternatively, if the gradient is that of the reward function, we can move in the direction of the gradient to maximize the expectancy of the rewards.

But there exists a problem in implementing this theory in practice. To understand this problem let us understand mathematically what we are trying to do. Let us take a policy denoted by π. This policy π parametrized over the parameter vector θ. The value of this policy π could be defined as the expectancy of the (discounted) cumulative rewards following this policy. Similar to symbols V for denoting state value and symbol Q denoting action value, we would use the symbol ‘J’ to denote the value of the ***Performance*** of policy π. The policy value J could be mathematically defined as equation (2) below π.

\[J_{(\theta)} = \mathbb{E} \sum_{t \geqslant 0} [\gamma^t r_t | \pi_\theta] \]-- (2)

$$J_{(\theta)} = \mathbb{E} \sum_{t \geqslant 0} [\gamma^t r_t | \pi_\theta]$$

Therefore, the most optimal parameter vector (θ*) will maximize this expected reward value following this policy, such as:

\[\theta^* = \argmax_\theta J_{(\theta)} \] --- (3)

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, J_{(\theta)}$$

Now let us introduce a new term, 'trajectory'. We will use the symbol τ (called 'tau') to denote 'trajectory'. The trajectory here refers to the sequence of states visited in an episode. In a stochastic policy the next action and hence the subsequently visited state need not be deterministic. Hence even under a given stochastic policy, different sequence of states could be visited with varying probabilities. Any such sequence of visited states (by taking some action and receiving the corresponding instantaneous rewards) at different time-steps in an episode is called a trajectory. The trajectory τ could be represented by $\tau = [(s_0, a_0, r_0); (s_1, a_1, r_1); \ldots; (s_t, a_t, r_t)]$.

The trajectory is influenced by the state-transition probabilities under a given policy. So, in terms of the trajectory, equation (2) for the policy value J could be re-written as equation (4) below:

\[J_{(\theta)} = \mathbb{E}_\tau \sum {\Pr(\tau;\theta)} r_{(\tau)} \]--- (4)

$$J_{(\theta)} = \mathbb{E}_\tau \sum \Pr(\tau;\theta) r_{(\tau)}$$

This equation simply says the Performance value of a given stochastic policy could be expressed as the expectancy of reward in a particular trajectory under a policy, weighed by the probability of attaining that trajectory under that policy. Since Expectancy could be integrated, so we further represent the above expression of expectancy over the different trajectories using integration (so that we could demonstrate the differentiation step easily) as:

\[J_{(\theta)} = \int_{\tau} r_{(\tau)} \Pr(\tau;\theta) d\tau \] --- (5)

$$J_{(\theta)} = \int_\tau r_{(\tau)} \Pr(\tau;\theta) d\tau$$

To obtain the gradient of the above expression, the function J needs to be differentiated with respect to the parameters θ as below:

\[\Delta_{\theta} J_{(\theta)} = \int_{\tau} r_{(\tau)} \Delta_{\theta} \Pr(\tau;\theta) d\tau \] --- (6)

$$\Delta_\theta J_{(\theta)} = \int_\tau r_{(\tau)} \Delta_\theta \Pr(\tau;\theta) d\tau$$

The equation (6) above is difficult to solve because of mathematical *'intractability'*. An *intractable* problem mathematics are the ones for which there exists no mathematical formulation to solve these problems efficiently. Equation (6) is intractable due to the fact that in the above equation we are trying to differentiate a function p (τ; θ) over the parameter (vector) θ when that function is itself conditioned (or depends) on the same parameter. So, in the exact mathematical solution it is difficult to implement policy-gradient method. Therefore, next we will learn about a very important algorithm called 'REINFORCE' and the associated mathematical simplification of the above equation to solve this problem.

The REINFORCE algorithm

The REINFORCE algorithm was proposed by Ronald J. Williams. Ronald also gave some mathematical simplification to the Equation (6) above so as to implement it in the proposed algorithm. Under limiting conditions, the differential of part of equation (6) causing intractability could be re-written as equation (7) below.

\[\Delta_{\theta} \Pr(\tau;\theta) d\tau = \Pr(\tau;\theta) \frac{\Delta_{\theta} \Pr(\tau;\theta)} {\Pr(\tau;\theta) } \] --- (7a)

$$\Delta_\theta \Pr(\tau;\theta) d\tau = \Pr(\tau;\theta) \frac{\Delta_\theta \Pr(\tau;\theta)}{\Pr(\tau;\theta)}$$

Or under limiting condition equation 7(a) could be rewritten as equation 7(b):

\[\Delta_{\theta} \Pr(\tau;\theta) d\tau = \Pr(\tau;\theta) \Delta_{\theta} \log {\Pr(\tau;\theta)} \] --- (7b)

$$\Delta_\theta \Pr(\tau;\theta) d\tau = \Pr(\tau;\theta) \Delta_\theta \log \Pr(\tau;\theta)$$

Therefore, re-writing equation (6) under this limiting form we have equation (8) below:

\[\Delta_{\theta} J_{(\theta)} = \int_{\tau} (r_{(\tau)} \Delta_{\theta}\log {\Pr(\tau;\theta)}) \Pr(\tau;\theta) d\tau \] --- (8)

$$\Delta_\theta J_{(\theta)} = \int_\tau (r_{(\tau)} \Delta_\theta \log \Pr(\tau;\theta)) \Pr(\tau;\theta) d\tau$$

Re-writing the equation (8) in Expectancy form, we get equation (9) as below:

\[\Delta_{\theta} J_{\theta} = \mathbb{E}_{\tau \sim \Pr(\tau;\theta)} [r_{\tau} \Delta_{\theta}\log {\Pr(\tau;\theta)}] \]--- (9)

$$\Delta_\theta J_\theta = \mathbb{E}_{\tau \sim \Pr(\tau;\theta)}[r_\tau \Delta_\theta \log \Pr(\tau;\theta)]$$

This is one part of the intended mathematical simplifications. In the above form the method could be implemented using 'Monte Carlo' sampling. In Monte Carlo sampling is we can simulate multiple experiments corresponding to a given policy and extract data from these experiments. REINFORCE method therefore could also be referred to as Monte Carlo approach to policy-gradient or simply ***'Monte-Carlo Policy Gradient'***. But there exists another practical (not necessarily mathematical) problem in implementing this yet. This is of knowing the P (τ; θ), that is the probability of different trajectories themselves under a given policy. Therefore, now even though we could solve this mathematically, but we have no efficient way to obtain the necessary trajectory probability in advance. This problem is also solved by slightly altering the mathematics of this equation as below.

\[\Pr(\tau;\theta) = \prod_{ t \geqslant 0} \Pr(s_{t+1} \mid s_t,a_t) \pi_{\theta}(a_t \given s_t) \] --- (10)

$$\mathbb{P}(\tau;\theta) = \prod_{t \geqslant 0} \mathbb{P}(s_{t+1} \mid s_t, a_t)\pi_\theta(a_t \mid s_t)$$

Since under Markov Decision Process (MDP), we assume that a given state once achieved is independent of previous happenings, so under this conditional independence assumption, we could essentially multiple the subsequent state-transition probabilities in a trajectory to obtain the overall trajectory probability as shown above in equation (10). Thus, the log p (τ; θ), could be rewritten as below:

\[\log {\Pr(\tau;\theta)} = \sum_{ t \geqslant 0} \log {\Pr(s_{t+1} \mid s_t,a_t)} + \log {\pi_{\theta}(a_t \mid s_t)} \] --- (11)

$$\log \mathbb{P}(\tau;\theta) = \sum_{t \geqslant 0} \log \mathbb{P}(s_{t+1} \mid s_t, a_t) + \log \pi_\theta(a_t \mid s_t)$$

Hence the differential of this formulation of log p (τ; θ) does not depend upon the trajectory probability distribution as earlier. It only depends on a series of state transition probability as we were using earlier in some of the other algorithms. This simplifies the equation (11) further and removes the requirement to trajectory probability as can be seen below in equation (12).

\[\Delta_{\theta}\log {\Pr(\tau;\theta)} = \sum_{ t \geqslant 0} \Delta_{\theta} \log {\pi_{\theta}(a_t \mid s_t)} \] --- (12)

$$\Delta_\theta \log \mathbb{P}(\tau;\theta) = \sum_{t \geqslant 0} \Delta_\theta \log \pi_\theta(a_t \mid s_t)$$

Therefore, rewriting the equation (9) for the gradient of policy value J to replace the trajectory probability distribution with the above form as in equation (12), we have the following equation (13) as below.

\[\Delta_{\theta} J_{(\theta)} \approx \sum_{t \geqslant 0} r_{(\tau)} \Delta_{\theta}\log {\pi_{\theta}(a_t \mid s_t)} \] --- (13)

$$\Delta_\theta J_{(\theta)} \approx \sum_{t \geqslant 0} r_{(\tau)} \Delta_\theta \log \pi_\theta(a_t \mid s_t)$$

Shortcomings of the REINFORCE algorithm

Even with the different mathematical simplifications and algorithmic enhancements, REINFORCE algorithm is not used in practice. This is because the gradient so obtained using the REINFORCE method has very ***high variance***.

One reason for such high variance is the form in which the rewards that are used in REINFORCE. In REINFORCE absolute rewards are used, and with each experiment of Monte Carlo the rewards may vary a lot. This leads to a very high variance in the so obtained gradient. But nevertheless, the two mathematical enhancements that we discussed, are very important to know and understand as most of the policy-gradient approaches use some part of REINFORCE algorithm especially the mathematical simplifications and develop further. Later we will discuss how some algorithms take these enhancements and slightly modify the computation to overcome this high variance problem.

Yet another reason for this high variance is the attribution of the reward to the specific state-action instances in the trajectory. Since REINFORCE in a sense averages out the rewards in a given trajectory, so if the rewards are due to some specific good state-action decisions only, and not because of most of the other state-actions in the same trajectory/ experiment, it becomes challenging to get the correct and specific attributions to those state-actions alone. This effect further leads to high variance.

Pseudo-Code for the REINFORCE algorithm

The above equations could be implemented in an iterative 'Monte-Carlo' like approach using the below pseudo-code.

function REINFORCE
Initialise θ arbitrarily
for each episode $\{(s_1, a_1, r_2), ...,(s_{T-1}, a_{T-1}, r_T)\} \sim \pi_\theta$ **do**
for $t = 1$ to $T - 1$ **do**
$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t , a_t) v_t$
end for
end for
return θ
end function

Where, v_t is the unbiased estimate sample of $Q_{\pi\theta}(s_t, a_t)$ and α is the step-size.

Methods to reduce variance in the REINFORCE algorithm

As we discovered in the earlier section on the shortcomings of the REINFORCE algorithm, the practicality of the REINFORCE algorithm is severely restricted because of the variance in the policy gradient. This high variance was mainly due to the fact that we were not able to deterministically and specifically identify which actions attributed to the reward in a given trajectory. This is turn meant that we were not able to positively move the gradient so that the subsequently updated policy could favour the best rewarding actions and restrict the gradient to discourage the actions which were not so rewarding. In this section we will discuss some approaches that we could use to overcome this problem in the REINFORCE algorithm.

It should be pointed out that some of these approaches to reduce variance that we will be discussing next and their subsequent variants are common to most of policy-gradient-based approaches and not just the REINFORCE algorithm. Hence the techniques that will be discussed are also the basis of some of the other algorithms that fall under 'policy-based approaches'. So, we would carry-forward

some part of this discussion to a later chapters when we would discuss those algorithms.

Cumulative Future Reward Based Attribution

We discussed earlier that the gradient of the policy-estimation function (policy-estimator) could be represented as:

\[\Delta_{\theta} J_{(\theta)} \approx \sum_{t \geqslant 0} r_{(\tau)} \Delta_{\theta}\log {\pi_{\theta}(a_t \mid s_t)} \] --- (13)

$$\Delta_\theta J_{(\theta)} \approx \sum_{t \geqslant 0} r_{(\tau)} \Delta_\theta \log \pi_\theta(a_t \mid s_t)$$

As we notice, in this form the attribution of reward (or penalties) that has been received in the past is also given to all the actions in the trajectory, even the actions that occur in the trajectory after receiving the specific instantaneous reward (future actions). That makes little sense.

For example, assume that in the grid-world example that we discussed earlier, one of the actions in a particular state made you lose 100 points (penalty) by moving to the ditch. Can this penalty be rightly attributed to an action in any other given state, say the last state just before you reached the 'treasure-state' that led you to the receive the 'treasure'? Obviously, such attribution does not seem correct.

Though we are not sure which specific future action could be attributed in which proportion to the subsequent future rewards, but we could at least say that any reward being realized before a particular action (in a particular state) could not be attributed to an action (as suggested by the policy) after realizing the reward. Therefore, we would slightly change the equation (13) to restrict the attribution of only future cumulative rewards to any current actions or particular policy decision (action in a particular state). This could be expressed as equation (14) below.

\[\Delta_{\theta} J_{(\theta)} \approx \sum_{t \geqslant 0} (\sum_{t' \geqslant t} r_{(\tau)}) \Delta_{\theta}\log {\pi_{\theta}(a_t \mid s_t)} \] --- (14)

$$\Delta_\theta J_{(\theta)} \approx \sum_{t \geqslant 0} (\sum_{t' \geqslant t} r_{(\tau)}) \Delta_\theta \log \pi_\theta(a_t \mid s_t)$$

Discounted Cumulative Future Rewards

As we discussed in the earlier section, changing the attribution from all rewards to all-future-rewards helps. We also discussed that we wanted to avoid equal attribution to all future actions (policy decisions). One way of doing this is by discounting the future rewards before attributing them to a given action. This ensures that the more recent actions/ policy-decisions get higher attribution for any upcoming rewards that the ones that occurred very early in the sequence. As we discussed earlier in this book, the intuition behind this is that more recent actions may better determine the realization of a reward, than something that has happened much earlier.

Modifying equation (14) to include the discounting parameter γ and raising it to the step/ time difference between the action and the reward realization, we get the equation (15) as below.

\[\Delta_{\theta} J_{(\theta)} \approx \sum_{t \geqslant 0} (\sum_{t' \geqslant t} \gamma^{t'-t} r_{(\tau)}) \Delta_{\theta}\log {\pi_{\theta}(a_t \mid s_t)} \] --- (15)

$$\Delta_\theta J_{(\theta)} \approx \sum_{t \geqslant 0} (\sum_{t' \geqslant t} \gamma^{t'-t} r_{(\tau)}) \Delta_\theta \log \pi_\theta(a_t \mid s_t)$$

REINFORCE with Baseline

In the present form as the equation (15) stands, we have the ability to move aggressively in the direction the gradient that ensures higher rewards, and conservatively in the direction of the gradient that ensures less rewards. Continuing with the same theme of identifying and attributing the rewards/ penalties to the correct actions, we need to start differentiating between better and worse rewards.

For illustration, let us take our grid-world example further. If from a particular state no matter where we move, we would get some positive reward, and the only thing that differs is either the quantum of reward or how-delayed the response is (the delay or late rewards in turn would get reflected as the quantum of reward as well if discounting is used), we could end-up moving in the direction of less favorable rewards as well (although slightly slow) along with moving in the direction of the more optimal rewards (although faster as compared to our movement in the direction of less rewarding actions). This is clearly counter-productive, as we intend to differentiate and identify the best action as lucidly and as soon as possible (from a training perspective). To ensure that we clearly identify the actions which we should we have concentrated moving only towards the most optimal reward, even moving away from actions that lesser rewards even if these are positive rewards. This is because these actions have the opportunity cost associated with them. In the same step that we took a less rewarding action, we could have instead taken a more rewarding action. So, we need to now start differentiating across the actions with respect to their quantum of rewards as well.

So as to achieve this objective, we need to understand how much an action is better or worse than a ***'baseline'*** scenario. This 'baseline' could simply be a constant. But as such a constant baseline does not optimally serve the purpose to positively rewarding the better actions and negatively reward the not-so-good actions. Reflecting back to our discussion on ***'advantage'*** in the 'Dueling DQN' algorithm, we had a similar discussion, and we discussed that the 'advantage function' helps us identify the relative advantage of a given action in a particular state over the base value of the state. We were thus able to differentiate quantifiably across different actions permissible in a state. So, essentially, we used the Value of the state as the ***'baseline'*** (value) for any action permissible in that state. We would do a similar thing here and take a baseline reward value for a particular state and take a difference of the actual (future and discounted) rewards received with this baseline to assess the goodness, or in this case also worseness (in case if the difference is negative) with this baseline. This could be implemented as equation (16) below.

\[\Delta_{\theta} J_{(\theta)} \approx \sum_{t \geqslant 0} (\sum_{t' \geqslant t} \gamma^{(t'- t)} r_{(\tau)} - b_{(s_t)}) \Delta_{\theta}\log {\pi_{\theta}(a_t \mid s_t) }\] --- (16)

$$\Delta_\theta J_{(\theta)} \approx \sum_{t \geqslant 0} (\sum_{t' \geqslant t} \gamma^{(t'-t)} r_{(\tau)} - b_{(s_t)}) \Delta_\theta \log \pi_\theta(a_t \mid s_t)$$

Choosing a baseline for the REINFORCE algorithm

In the previous section, we identified that using a baseline reward value for each state may help distinguish a good action from a better one or a worse one, and hence also help in reducing the variance of the gradient of the REINFORCE method. We briefly also discussed the 'advantage' function, that we dealt in the 'Dueling DQN' algorithm. But in a policy-gradient based REINFORCE we do not want to get into the state-value V(s) calculation (for now) and want a simpler baseline that can reflect some good indication of average (future, discounted) rewards from a particular state.

To achieve this one could come up with different types of baseline. There are some stated in different literature. One of them could be a using a constant moving average of the cumulative future discounted rewards received from all the trajectories which pass through this particular state. Since such rewards are computed in the REINFORCE algorithm itself, we don't need an external value computation function to enable this, and the resulting implementation could be simple and straight forward.

Yet another, more expressive baseline could be to take the actual advantage of a given action in a particular state, which is represented by the difference between the Q value of that action in that state and the State Value V of that state as (Q (s, a) – V(s)). Also, since Q and V value already have a sense of reward in-built in them and a good treatment for differentiating across different types of future rewards and discounted rewards, we need not bother about including such factors separately. Therefore, the equation (16) could be modified to replace the explicit reward and baseline as a composite reward with baseline using Q and V functions as equation (17) below

\[\Delta_{\theta} J_{(\theta)} \approx \sum_{t \geqslant 0} (Q_{\pi_{\theta},(s_t, a_t)} - V_{\pi_{\theta},(s_t)}) \Delta_{\theta}\log {\pi_{\theta}(a_t \mid s_t) } \] --- (17)

$$\Delta_\theta J_{(\theta)} \approx \sum_{t \geqslant 0} (Q_{\pi_\theta,(s_t,a_t)} - V_{\pi_\theta,(s_t)}) \Delta_\theta \log \pi_\theta(a_t \mid s_t)$$

Summary

As opposed to value-based approaches, in which although the value estimates are stochastic, but the policy itself is deterministic, the policy-based approaches provides a mechanism to have a stochastic policy. Policy-based approaches are more direct as we are directly exploring the policy that takes action-decisions instead of first estimating the values and then forming a policy based on such value estimates. Policy-based approaches also provides better mechanism to work with scenarios having large action-space, and can also work with scenarios that requires continuous action control.

But it is difficult to implement a policy-based approach, especially the policy-gradient based approaches. One reason for this is that the mathematics for finding the gradient of the policy value is intractable. With some simplifications and limiting assumptions we were able to overcome this issue, and implement in a Monte Carlo simulation-based algorithm called REINFORCE. But the resulting format of mathematical simplifications leads to a solution that has a lot of variance in gradient computation.

With some further assumptions and techniques leading to correct reward attribution for future rewards and reward baseline we were able to bring down the variance in gradient computation in the REINFROCE algorithm. Such enhancements are not just limited to the REINFORCE algorithm, but will go a long way forward in making other advanced policy-based approaches more practical to implement thereby opening the gates for more powerful Reinforcement Learning for real-life scenarios having large action-space to work with and for scenarios requiring continuous actions control.

References

Richard S. Sutton, David McAllester, Satinder Singh, Yishay Mansour, "Policy Gradient Methods for Reinforcement Learning with Function Approximation", Proceedings of the 12th International Conference on Neural Information Processing Systems, NIPS'99, pp. 1057-1063, 1999.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, Martin Riedmiller, "Deterministic Policy Gradient Algorithms", Proceedings of the 31st International Conference on Machine Learning, pp. 387-395, 2014.

David Silver, "Lecture 7 - Policy Gradient", University College London, url: http://www0.cs.ucl.ac.uk/ staff/d.silver/web/Teaching_files/pg.pdf, Accessed: Aug, 2018.

Fei-Fei Li & Justin Johnson & Serena Yeung, “Lecture 14”, CS231 - Stanford, url: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture14.pdf, Accessed: Aug, 2018.

Ronald J. Williams. A class of gradient-estimating algorithms for reinforcement learning in neural networks. In Proceedings of the IEEE First International Conference on Neural Networks, San Diego, CA, 1987.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning, 8(3):229-256, 1992.