Going Deeper Into Reinforcement Learning: Fundamentals of Policy Gradients

Mar 27, 2017

As I stated in my last blog post, I am feverishly trying to read more research papers. One category of papers that seems to be coming up a lot recently are those about *policy gradients*, which are a popular class of reinforcement learning algorithms which estimate a gradient for a function approximator. Thus, the purpose of this blog post is for me to explicitly write the mathematical foundations for policy gradients so that I can gain understanding. In turn, I hope some of my explanations will be useful to a broader audience of AI students.

Assumptions and Problem Statement

In any type of research domain, we always have to make some set of assumptions. (By “we”, I refer to the researchers who write papers on this.) With reinforcement learning and policy gradients, the assumptions usually mean the **episodic** setting where an agent engages in multiple **trajectories** in its environment. As an example, an agent could be playing a game of Pong, so one episode or trajectory consists of a full start-to-finish game.

We define a trajectory τ of length T as

$$\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$$

where s_0 comes from the starting distribution of states, $a_i \sim \pi_\theta(a_i|s_i)$, and $s_i \sim P(s_i|s_{i-1}, a_{i-1})$ with P the dynamics model (i.e. how the environment changes). We actually *ignore* the dynamics when optimizing, since all we care about is getting a good gradient signal for π_θ to make it better. If this isn’t clear now, it will be clear soon. Also, the reward can be computed from the states and actions, since it’s usually a function of (s_i, a_i, s_{i+1}), so it’s not technically needed in the trajectory.

What’s our *goal* here with policy gradients? Unlike algorithms such as DQN, which strive to find an excellent policy indirectly through Q-values, policy gradients perform a *direct* gradient update on a policy to change its parameters, which is what makes it so appealing. Formally, we have:

$$\text{maximize}_\theta \; \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} \gamma^t r_t\right]$$

- **Note I**: I put π_θ under the expectation. This means the rewards are computed from a trajectory which was generated under the policy π_θ. We have to *find* “optimal” settings of θ to make this work.

- **Note II**: we don't need to optimize the expected sum of discounted rewards, though it's the formulation I'm most used to. Alternatives include ignoring γ by setting it to one, extending T to infinity if the episodes are infinite-horizon, and so on.

The above raises the all-important question: *how do we find the best* θ? If you've taken optimization classes before, you should know the answer already: perform gradient ascent on θ, so we have $\theta \leftarrow \theta + \alpha \nabla f(x)$ where $f(x)$ is the function being optimized. Here, that's the expected value of whatever sum of rewards formula we're using.

Two Steps: Log-Derivative Trick and Determining Log Probability

Before getting to the computation of the gradient, let's first review two mathematical facts which will be used later, and which are also of independent interest. The first is the "log-derivative" trick, which tells us how to insert a log into an expectation when starting from $\nabla_\theta \mathbb{E}[f(x)]$. Specifically, we have:

$$\begin{aligned}
\nabla_\theta \mathbb{E}[f(x)] &= \nabla_\theta \int p_\theta(x) f(x) dx \\
&= \int \frac{p_\theta(x)}{p_\theta(x)} \nabla_\theta p_\theta(x) f(x) dx \\
&= \int p_\theta(x) \nabla_\theta \log p_\theta(x) f(x) dx \\
&= \mathbb{E}\Big[f(x) \nabla_\theta \log p_\theta(x) \Big]
\end{aligned}$$

where p_θ is the density of x. Most of these steps should be straightforward. The main technical detail to worry about is exchanging the gradient with the integral. I have never been comfortable in knowing when we are allowed to do this or not, but since everyone else does this, I will follow them.

Another technical detail we will need is the gradient of the log probability of a *trajectory* since we will later switch x from above with a trajectory τ. The computation of $\log p_\theta(\tau)$ proceeds as follows:

$$\begin{aligned}
\nabla_\theta \log p_\theta(\tau) &= \nabla \log \left(\mu(s_0) \prod_{t=0}^{T-1} \pi_\theta(a_t|s_t) P(s_{t+1}|s_t, a_t) \right) \\
&= \nabla_\theta \left[\log \mu(s_0) + \sum_{t=0}^{T-1} (\log \pi_\theta(a_t|s_t) + \log P(s_{t+1}|s_t, a_t)) \right] \\
&= \nabla_\theta \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t)
\end{aligned}$$

The probability of τ decomposes into a chain of probabilities by the Markov Decision Process assumption, whereby the next action only depends on the current state, and the next state only depends on the current state and action. To be explicit, we use the functions that we already defined: π_θ and P for the policy and dynamics, respectively. (Here, μ represents the starting state distribution.) We also observe that when taking gradients, the dynamics disappear!

Computing the Raw Gradient

Using the two tools above, we can now get back to our original goal, which was to compute the gradient of the expected sum of (discounted) rewards. Formally, let $R(\tau)$ be the reward function we want to optimize (i.e. maximize). Using the above two tricks, we obtain:

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

In the above, the expectation is with respect to the policy function, so think of it as $\tau \sim \pi_\theta$. In practice, we need trajectories to get an empirical expectation, which estimates this actual expectation.

So that's the gradient! Unfortunately, we're not quite done yet. The naive way is to run the agent on a batch of episodes, get a set of trajectories (call it $\hat{\tau}$) and update with $\theta \leftarrow \theta + \alpha\nabla_\theta\mathbb{E}_{\tau\in\hat{\tau}}[R(\tau)]$ using the empirical expectation, but this will be too slow and unreliable due to high variance on the gradient estimates. After one batch, we may exhibit a wide range of results: much better performance, equal performance, or *worse* performance. The high variance of these gradient estimates is precisely why there has been so much effort devoted to variance reduction techniques. (I should also add from personal research experience that variance reduction is certainly not limited to reinforcement learning; it also appears in many statistical projects which concern a bias-variance tradeoff.)

How to Introduce a Baseline

The standard way to reduce the variance of the above gradient estimates is to insert a **baseline function** $b(s_t)$ inside the expectation.

For concreteness, assume $R(\tau) = \sum_{t=0}^{T-1} r_t$, so we have no discounted rewards. We can express the policy gradient in three equivalent, but perhaps non-intuitive ways:

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}\Big[R(\tau)\Big] &\overset{(i)}{=} \mathbb{E}_{\tau\sim\pi_\theta}\left[\left(\sum_{t=0}^{T-1} r_t\right)\cdot\nabla_\theta\left(\sum_{t=0}^{T-1}\log\pi_\theta(a_t|s_t)\right)\right] \\
&\overset{(ii)}{=} \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t'=0}^{T-1} r_{t'}\sum_{t=0}^{t'}\nabla_\theta\log\pi_\theta(a_t|s_t)\right] \\
&\overset{(iii)}{=} \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1}\nabla_\theta\log\pi_\theta(a_t|s_t)\left(\sum_{t'=t}^{T-1} r_{t'}\right)\right]
\end{aligned}$$

Comments:

- **Step (i)** follows from plugging in our chosen $R(\tau)$ into the policy gradient we previously derived.
- **Step (ii)** follows from first noting that $\nabla_\theta\mathbb{E}_\tau\Big[r_{t'}\Big] = \mathbb{E}_\tau\left[r_{t'}\cdot\sum_{t=0}^{t'}\nabla_\theta\log\pi_\theta(a_t|s_t)\right]$. The reason why this is true can be somewhat tricky to identify. I find it easy to think of just re-defining $R(\tau)$ as $r_{t'}$ for some fixed time-step t'. Then, we do the exact same computation above to get the final result, as shown in the equation of the "Computing the Raw Gradient" section. The main difference now is that since we're considering the reward at time t', our trajectory under expectation *stops* at that time. More concretely, $\nabla_\theta\mathbb{E}_{(s_0,a_0,\ldots,s_T)}\Big[r_{t'}\Big] = \nabla_\theta\mathbb{E}_{(s_0,a_0,\ldots,s_{t'})}\Big[r_{t'}\Big]$. This is like "throwing

away variables" when taking expectations due to "pushing values" through sums and summing over densities (which cancel out); I have another example later in this post which makes this explicit.

Next, we sum over both sides, for $t' = 0, 1, \ldots, T-1$. Assuming we can exchange the sum with the gradient, we get

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}\left[R(\tau)\right] &= \nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t'=0}^{T-1} r_{t'}\right] \\
&= \sum_{t'=0}^{T-1} \nabla_\theta \mathbb{E}_{\tau^{(t')}}\left[r_{t'}\right] \\
&= \sum_{t'}^{T-1} \mathbb{E}_{\tau^{(t')}}\left[r_{t'} \cdot \sum_{t=0}^{t'} \nabla_\theta \log \pi_\theta(a_t|s_t)\right] \\
&= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t'}^{T-1} r_{t'} \cdot \sum_{t=0}^{t'} \nabla_\theta \log \pi_\theta(a_t|s_t)\right].
\end{aligned}$$

where $\tau^{(t')}$ indicates the trajectory up to time t'. (Full disclaimer: I'm not sure if this formalism with τ is needed, and I think most people would do this computation without worrying about the precise expectation details.)

- **Step (iii)** follows from a nifty algebra trick. To simplify the subsequent notation, let $f_t := \nabla_\theta \log \pi_\theta(a_t|s_t)$. In addition, **ignore the expectation**; we'll only re-arrange the inside here. With this substitution and setup, the sum inside the expectation from **Step (ii)** turns out to be

$$\begin{array}{c}
r_0 f_0 + \\
r_1 f_0 + r_1 f_1 + \\
r_2 f_0 + r_2 f_1 + r_2 f_2 + \\
\cdots \\
r_{T-1} f_0 + r_{T-1} f_1 + r_{T-1} f_2 \cdots + r_{T-1} f_{T-1}
\end{array}$$

  In other words, each $r_{t'}$ has its own *row* of f-value to which it gets distributed. Next, *switch to the column view*: instead of summing row-wise, sum *column-wise*. The first column is $f_0 \cdot \left(\sum_{t=0}^{T-1} r_t\right)$. The second is $f_1 \cdot \left(\sum_{t=1}^{T-1} r_t\right)$. And so on. Doing this means we get the desired formula after replacing f_t with its real meaning and hitting the expression with an expectation.

Note: it is *very easy* to make a typo with these. I checked my math carefully and cross-referenced it with references online (which *themselves* have typos). If any readers find a typo, please let me know.

Using the above formulation, we finally introduce our baseline b, which is a function of s_t (and *not* $s_{t'}$, I believe). We "insert" it inside the term in parentheses:

$$\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)] = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t)\right)\right]$$

At first glance, it doesn't seem like this will be helpful, and one might wonder if this would cause the gradient estimate to become biased. Fortunately, it turns out that this is not a problem. This was surprising to me, because all we know is that $b(s_t)$ is a function of s_t. However, this is a bit misleading because usually we want $b(s_t)$ to be the *expected return* starting at time t, which means it really "depends" on the subsequent time steps. For now, though, just think of it as a function of s_t.

Understanding the Baseline

In this section, I first go over why inserting b above doesn't make our gradient estimate biased. Next, I will go over why the baseline reduces variance of the gradient estimate. These two capture the best of both worlds: staying unbiased and reducing variance. In general, any time you have an unbiased estimate and it remains so after applying a variance reduction technique, then apply that variance reduction!

First, let's show that the gradient estimate is unbiased. We see that with the baseline, we can distribute and rearrange and get:

$$\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)] = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\left(\sum_{t'=t}^{T-1} r_{t'}\right) - \sum_{t=0}^{T-1}\nabla_\theta \log\pi_\theta(a_t|s_t) b(s_t)\right]$$

Due to linearity of expectation, all we need to show is that for any single time t, the gradient of $\log \pi_\theta(a_t|s_t)$ multiplied with $b(s_t)$ is zero. This is true because

$$\begin{aligned}
\mathbb{E}_{\tau\sim\pi_\theta}\Big[\nabla_\theta \log\pi_\theta(a_t|s_t)b(s_t)\Big] &= \mathbb{E}_{s_{0:t},a_{0:t-1}}\Big[\mathbb{E}_{s_{t+1:T},a_{t:T-1}}[\nabla_\theta \log\pi_\theta(a_t|s_t)b(s_t)]\Big] \\
&= \mathbb{E}_{s_{0:t},a_{0:t-1}}\Big[b(s_t)\cdot \underbrace{\mathbb{E}_{s_{t+1:T},a_{t:T-1}}[\nabla_\theta \log\pi_\theta(a_t|s_t)]}_{E}\Big] \\
&= \mathbb{E}_{s_{0:t},a_{0:t-1}}\Big[b(s_t)\cdot \mathbb{E}_{a_t}[\nabla_\theta \log\pi_\theta(a_t|s_t)]\Big] \\
&= \mathbb{E}_{s_{0:t},a_{0:t-1}}\Big[b(s_t)\cdot 0\Big] = 0
\end{aligned}$$

Here are my usual overly-detailed comments (apologies in advance):

- **Note I**: this notation is similar to what I had before. The trajectory $s_0, a_0, \ldots, a_{T-1}, s_T$ is now represented as $s_{0:T}, a_{0:T-1}$. In addition, the expectation is split up, which is allowed. If this is confusing, think of the definition of the expectation with respect to at least two variables. We can write brackets in any appropriately enclosed location. Furthermore, we can "omit" the un-necessary variables in going from $\mathbb{E}_{s_{t+1:T},a_{t:T-1}}$ to $\mathbb{E}_{a_t}$ (see expression E above). Concretely, assuming we're in discrete-land with actions in $\mathcal{A}$ and states in $\mathcal{S}$, this is because E evaluates to:

$$\begin{aligned}
E &= \sum_{a_t\in\mathcal{A}}\sum_{s_{t+1}\in\mathcal{S}}\cdots\sum_{s_T\in\mathcal{S}} \underbrace{\pi_\theta(a_t|s_t)P(s_{t+1}|s_t,a_t)\cdots P(s_T|s_{T-1},a_{T-1})}_{p((a_t,s_{t+1},a_{t+1},\ldots,a_{T-1},s_T))}(\nabla_\theta \log\pi_\theta(a_t|s_t)) \\
&= \sum_{a_t\in\mathcal{A}} \pi_\theta(a_t|s_t)\nabla_\theta \log\pi_\theta(a_t|s_t) \sum_{s_{t+1}\in\mathcal{S}} P(s_{t+1}|s_t,a_t) \sum_{a_{t+1}\in\mathcal{A}}\cdots\sum_{s_T\in\mathcal{S}} P(s_T|s_{T-1},a_{T-1}) \\
&= \sum_{a_t\in\mathcal{A}} \pi_\theta(a_t|s_t)\nabla_\theta \log\pi_\theta(a_t|s_t)
\end{aligned}$$

This is true because of the definition of expectation, whereby we get the joint density over the entire trajectory, and then we can split it up like we did earlier with the gradient of the log probability computation. We can distribute $\nabla_\theta \log \pi_\theta(a_t|s_t)$ all the way back to (but not beyond) the first sum over a_t. Pushing sums "further back" results in a bunch of sums over densities, each of which sums to one. The astute reader will notice that this is precisely what happens with variable elimination for graphical models. (The more technical reason why "pushing values back through sums" is allowed has to do with abstract algebra properties of the sum function, which is beyond the scope of this post.)

- **Note II**: This proof above also works with an infinite-time horizon. In Appendix B of the *Generalized Advantage Estimation* paper (arXiv link), the authors do so with a proof exactly matching the above, except that T and $T-1$ are now infinity.

- **Note III**: About the expectation going to zero, that's due to a well-known fact about *score* functions, which are precisely the gradient of log probailities. We went over this in my STAT 210A class last fall. It's *again* the log derivative trick. Observe that:

$$\mathbb{E}_{a_t}\Big[\nabla_\theta \log \pi_\theta(a_t|s_t)\Big] = \int \frac{\nabla_\theta \pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)} \pi_\theta(a_t|s_t) da_t = \nabla_\theta \int \pi_\theta(a_t|s_t) da_t = \nabla_\theta \cdot 1 = 0$$

where the penultimate step follows from how π_θ is a density. This follows for all time steps, and since the gradient of the log gets distributed for each t, it applies in all time steps. I switched to the continuous-land version for this, but it also applies with sums, as I just recently used in Note I.

The above shows that introducing b doesn't cause bias.

The last thing to cover is why its introduction reduces variance. I provide an approximate argument. To simplify notation, set $R_t(\tau) = \sum_{t'=t}^{T-1} r_{t'}$. We focus on the *inside* of the expectation (of the gradient estimate) to analyze the variance. The technical reason for this is that expectations are technically *constant* (and thus have variance zero) but in practice we have to approximate the expectations with trajectories, and that has high variance.

The variance is approximated as:

$$\begin{aligned}
{\rm Var}\left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)(R_t(\tau) - b(s_t))\right) &\overset{(i)}{\approx} \sum_{t=0}^{T-1} \mathbb{E}\tau \left[\Big(\nabla_\theta \log \pi_\theta(a_t|s_t)(R_t(\tau) - b(s_t))\Big)^2\right] \\
&\overset{(ii)}{\approx} \sum_{t=0}^{T-1} \mathbb{E}_\tau \left[\Big(\nabla_\theta \log \pi_\theta(a_t|s_t)\Big)^2\right] \mathbb{E}_\tau \left[\Big(R_t(\tau) - b(s_t)\Big)^2
\end{aligned}$$

Approximation (i) is because we are approximating the variance of a sum by computing the sum of the variances. This is not true in general, but if we can assume this, then by the definition of the variance ${\rm Var}(X) := \mathbb{E}[X^2] - (\mathbb{E}[X])^2$, we are left with the $\mathbb{E}[X^2]$ term since we already showed that introducing the baseline doesn't cause bias. **Approximation (ii)** is because we assume independence among the values involved in the expectation, and thus we can factor the expectation.

Finally, we are left with the term $\mathbb{E}_\tau \left[\Big(R_t(\tau) - b(s_t)\Big)^2\right]$. If we are able to optimize our choice of $b(s_t)$, then this is a least squares problem, and it is well known that the optimal choice of $b(s_t)$ is to be the expected value of $R_t(\tau)$. In fact, that's *why* policy gradient researchers usually want $b(s_t) \approx \mathbb{E}[R_t(\tau)]$ to approximate the expected return starting at time t, and that's *why* in the vanilla policy gradient algorithm we have to re-fit the baseline estimate each time to make it as close to the expected return $\mathbb{E}[R_t(\tau)]$. At last, I understand.

How accurate are these approximations in practice? My intuition is that they are actually fine, because recent advances in reinforcement learning algorithms, such as A3C, focus on the problem of breaking correlation among samples. If the correlation among samples is broken, then Approximation (i) becomes better, because I think the samples $s_0, a_0, \dots, a_{T-1}, s_T$ are *no longer generated from the same trajectory*.

Well, that's my intuition. If anyone else has a better way of describing it, feel free to let me know in the comments or by email.

Discount Factors

So far, we have assumed we wanted to optimize the expected return, or the expected *sum of rewards*. However, if you've studied value iteration and policy iteration, you'll remember that we usually use *discount factors* $\gamma \in (0, 1]$. These empirically work well because the effect of an action many time steps later is likely to be negligible compared to other action. Thus, it may not make sense to try and include raw distant rewards in our optimization problem. Thus, we often impose a discount as follows:

$$\begin{aligned} \nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)] &= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t)\right)\right] \\ &\approx \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \left(\sum_{t'=t}^{T-1} \gamma^{t'-t} r_{t'} - b(s_t)\right)\right] \end{aligned}$$

where the $\gamma^{t'-t}$ serves as the discount, starting from 1, then getting smaller as time passes. (The first line above is a repeat of the policy gradient formula that I describe earlier.) As this is not exactly the "desired" gradient, this is an *approximation*, but it's a reasonable one. This time, we now want our baseline to satisfy $b(s_t) \approx \mathbb{E}[r_t + \gamma r_{t+1} + \cdots + \gamma^{T-1-t} r_{T-1}]$.

Advantage Functions

In this final section, we replace the policy gradient formula with the following *value* functions:

$$Q^\pi(s,a) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} r_t \;\middle|\; s_0 = s, a_0 = a\right]$$

$$V^\pi(s) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} r_t \;\middle|\; s_0 = s\right]$$

Both of these should be familiar from basic AI; see the CS 188 notes from Berkeley if this is unclear. There are also *discounted* versions, which we can denote as $Q^{\pi,\gamma}(s,a)$ and $V^{\pi,\gamma}(s)$. In addition, we can also consider starting at any given time step, as in $Q^{\pi,\gamma}(s_t,a_t)$ which provides the expected (discounted) return assuming that at time t, our state-action pair is (s_t, a_t).

What might be new is the *advantage* function. For the undiscounted version, it is defined simply as:

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$$

with a similar definition for the discounted version. Intuitively, the advantage tells us how much better action a would be compared to the return based on an "average" action.

The above definitions look very close to what we have in our policy gradient formula. In fact, we can claim the following:

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)] &= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t)\right)\right] \\
&\overset{(i)}{=} \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot \Big(Q^\pi(s_t,a_t) - V^\pi(s_t)\Big)\right] \\
&\overset{(ii)}{=} \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot A^\pi(s_t,a_t)\right] \\
&\overset{(iii)}{\approx} \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot A^{\pi,\gamma}(s_t,a_t)\right]
\end{aligned}$$

In (i), we replace terms with their expectations. This is *not* generally valid to do, but it should work in this case. My guess is that if you start from the second line above (after the "(i)") and plug in the definition of the expectation inside and rearrange terms, you can get the first line. However, I have not had the time to check this in detail and it takes a lot of space to write out the expectation fully. The conditioning with the value functions makes it a bit messy and thus the law of iterated expectation may be needed.

Also from line (i), we notice that *the value function is a baseline*, and hence we can add it there without changing the unbiased-ness of the expectation. Then lines (ii) and (iii) are just for the advantage function. The implication of this formula is that the problem of policy gradients, in some sense, *reduces to finding good estimates* $\hat{A}^{\pi,\gamma}(s_t,a_t)$ *of the advantage function* $A^{\pi,\gamma}(s_t,a_t)$. That is precisely the topic of the paper *Generalized Advantage Estimation*.

Concluding Remarks

Hopefully, this is a helpful, self-contained, bare-minimum introduction to policy gradients. I am trying to learn more about these algorithms, and going through the math details is helpful. This will also make it easier for me to understand the increasing number of research papers that are using this notation.

I also have to mention: I remember a few years ago during the first iteration of CS 294-112 that I had no idea how policy gradients worked. Now, I think I have become slightly more enlightened.

Acknowledgements: I thank John Schulman for making his notes publicly available.

Update April 19, 2017: I have code for vanilla policy gradients in my reinforcement learning GitHub repository.

Seita's Place
seita@cs.berkeley.edu

DanielTakeshi
(Never!)

This is my blog, where I have written over 300 articles on a variety of topics. Recent posts tend to focus on computer science, my area of specialty as a Ph.D. student at UC Berkeley.