Reinforcement Learning Tutorial 6, Week 7
—
Policy Gradients: The REINFORCE algorithm[*] and Hyperparameters in RL

Pavlos Andreadis

March 2024

Overview: The following tutorial questions relate to material taught in weeks 5 and 6 of the 2023-24 Reinforcement Learning course. They aim at encouraging engagement with the course material and facilitating a deeper understanding.

We continue on the "AI controlled orchard" problem from tutorial 6 for a deep dive into a simple application of the REINFORCE algorithm. As you apply the algorithm, consider the changes to your policy and what they reflect. Also, consider the limitations in the suggested formulation of the problem (you are implicitly given that you have to use two state-action features, which you will have to define).

The second problem asks you to consider the use of the use of the learning rate and discount factor in Reinforcement Learning.

Problem 1 - Policy Gradients: REINFORCE

Consider the orchard problem from our last tutorial Andreadis [2021], and the trajectory representing its last harvest:

Concentration of A (ppm)	Concentration of B (ppm)	Concentration of C (ppm)	Action Taken	Profit/Reward (credits)
6	7	2	`Wait`	-1
0	5	2	`Wait`	-1
3	8	4	`Harvest`	19

[*]with special thanks to **Ross McKenzie** for introducing a first version of problem 1

Assume that these actions were taken using a policy parameterization with soft-max in action preferences with linear action preferences, and a known parameter $\theta_0 = [0, 0]$ (i.e. assume the state-action space is parameterised by two features, here chosen to be defined by the action *only*). Using the REINFORCE algorithm with a step size of $\alpha = 10^{-4}$, update your policy given the above trajectory.

By defining $\theta_0 = [0, 0]$ the exercise implies the use of 2 features but does not specify them. Let us define each state-action pair as:

$\boldsymbol{x}(s, a = \texttt{Harvest}) = [0, 1]$
$\boldsymbol{x}(s, a = \texttt{Wait}) = [1, 0]$

[*If you have some experience of Decision Theory* [Braziunas, 2006] (*or Recommender Systems*), *you can think of action preferences as a utility function over actions. The policy function then, to continue the metaphor, would be the behaviour model.*]

For Linear action preferences we have $h(s, a, \theta) = \theta^T \boldsymbol{x}(s, a)$. Our policy at time step 0 is:

$\pi_0(a|s, \boldsymbol{\theta}) = \frac{e^{h(s,a,\boldsymbol{\theta})}}{\sum_b e^{h(s,b,\boldsymbol{\theta})}} = \frac{e^{[0,0]^T \boldsymbol{x}(s,a)}}{\sum_b e^{[0,0]^T \boldsymbol{x}(s,b)}} = \frac{e^0}{e^0 + e^0} = \frac{1}{2}.$

[*Note that our state vector, in this case, only depends on the action taken but that this is not generally the case.*]

As we are taking the previous tutorial's setup, and the episodes terminate, we will set $\gamma = 1$. Since REINFORCE is a Monte Carlo algorithm, the target at each time-step is the full return:

$G_0 = \sum_{k=t+1=1}^{T=3} \gamma \cdot R_k = 1 \cdot R_1 + 1 \cdot R_2 + 1 \cdot R_3 = -1 - 1 + 19 = 17.$

This gives us:

$$\begin{aligned}
\boldsymbol{\theta}_1 &= \boldsymbol{\theta}_0 \; + \; \alpha \, \gamma \, G_0 \; \nabla ln \, \pi(A_0|S_0, \boldsymbol{\theta}_0) \\
&= \boldsymbol{\theta}_0 \; + \; \alpha \, \gamma \, G_0 \, \Big(\boldsymbol{x}(S_0, A_0) - \sum_b \pi(b|S_0, \boldsymbol{\theta}_0) \, \boldsymbol{x}(S_0, b) \Big) \\
&= [0, 0] \; + \; 10^{-4} \cdot 1 \cdot 17 \cdot \Big([1, 0] - (\frac{1}{2}[0, 1] + \frac{1}{2}[1, 0]) \Big) \\
&= [0, 0] + 10^{-4} \cdot 17 \cdot \Big[\frac{1}{2}, -\frac{1}{2} \Big] \\
&= [0.85 \cdot 10^{-3}, -0.85 \cdot 10^{-3}].
\end{aligned}$$

Proceeding similarly, we have:

$G_1 = \sum_{k=2}^{T=3} \gamma \cdot R_k = R_2 + R_3 = -1 + 19 = 18,$

and the policy

$$\begin{aligned}
\pi_1(a = \texttt{Harvest}|s, \boldsymbol{\theta}_1) &= \frac{e^{[0.85\cdot 10^{-3}, -0.85\cdot 10^{-3}]^T [0,1]}}{\sum_b e^{[0.85\cdot 10^{-3}, -0.85\cdot 10^{-3}]^T \boldsymbol{x}(s,b)}} = \frac{e^{-0.85\cdot 10^{-3}}}{e^{-0.85\cdot 10^{-3}} + e^{0.85\cdot 10^{-3}}} \\
&\approx \frac{(1/1.00085)}{(1/1.00085) + 1.00085} \approx \frac{0.99915}{2} \approx 0.49957 \\
\pi_1(a = \texttt{Wait}|s, \boldsymbol{\theta}_1) &= \frac{e^{[0.85\cdot 10^{-3}, -0.85\cdot 10^{-3}]^T [1,0]}}{\sum_b e^{[0.85\cdot 10^{-3}, -0.85\cdot 10^{-3}]^T \boldsymbol{x}(s,b)}} = \frac{e^{0.85\cdot 10^{-3}}}{e^{-0.85\cdot 10^{-3}} + e^{0.85\cdot 10^{-3}}} \\
&\approx \frac{1.0008)}{(1/1.00085) + 1.00085} \approx 1 - 0.49957 = 0.50043.
\end{aligned}$$

Giving us the update:

$$\begin{aligned}
\boldsymbol{\theta}_2 &= \boldsymbol{\theta}_1 \ + \ \alpha\, \gamma\, G_1\, \nabla ln\, \pi(A_1|S_1, \boldsymbol{\theta}_1) \\
&= \boldsymbol{\theta}_1 \ + \ \alpha\, \gamma\, G_1 \Big(\boldsymbol{x}(S_1, A_1) - \sum_b \pi(b|S_1, \boldsymbol{\theta}_1)\, \boldsymbol{x}(S_1, b) \Big) \\
&= [0.85\cdot 10^{-3}, -0.85\cdot 10^{-3}] \ + \ 10^{-4}\cdot 1\cdot 18\cdot \Big([1,0] - (0.49957\,[0,1] + 0.50043\,[1,0]) \Big) \\
&= [0.85\cdot 10^{-3}, -0.85\cdot 10^{-3}] + 10^{-4}\cdot 18\cdot [0.49957, -0.49957] \\
&\approx [1.75\cdot 10^{-3}, -1.75\cdot 10^{-3}].
\end{aligned}$$

[*Note how we are computing the probability of taking each action under the updated policy. This is not the policy we used to sample our trajectory from. As REINFORCE is a control algorithm, the idea is that you will sample the next trajectory using the updated policy from the end of the previous trajectory.*]

And finally:

$G_2 = \sum_{k=3}^{T=3} \gamma \cdot R_k = R_3 = 19 = 19,$

and

$$\begin{aligned}
\pi_2(a = \texttt{Harvest}|s, \boldsymbol{\theta}_2) &= \frac{e^{[1.75\cdot 10^{-3}, -1.75\cdot 10^{-3}]^T [0,1]}}{\sum_b e^{[1.75\cdot 10^{-3}, -1.75\cdot 10^{-3}]^T \boldsymbol{x}(s,b)}} \\
&= \frac{e^{-1.75\cdot 10^{-3}}}{e^{-1.75\cdot 10^{-3}} + e^{1.75\cdot 10^{-3}}} \\
&\approx \frac{0.998}{0.998 + 1.002} \approx 0.499 \\
\pi_2(a = \texttt{Wait}|s, \boldsymbol{\theta}_2) &= \frac{e^{[1.75\cdot 10^{-3}, -1.75\cdot 10^{-3}]^T [1,0]}}{\sum_b e^{[1.75\cdot 10^{-3}, -1.75\cdot 10^{-3}]^T \boldsymbol{x}(s,b)}} \\
&= \frac{e^{1.75\cdot 10^{-3}}}{e^{-1.75\cdot 10^{-3}} + e^{1.75\cdot 10^{-3}}} \\
&\approx 1 - 0.499 = 0.501.
\end{aligned}$$

[*Note how the probability assigned to harvesting is decreasing. Why is this?*]

With the final update for this trajectory (where we are harvesting):

$$\begin{aligned}
\boldsymbol{\theta}_3 &= \boldsymbol{\theta}_2 \;+\; \alpha\,\gamma\,G_2\;\nabla ln\,\pi(A_2|S_2,\boldsymbol{\theta}_2)\\
&= \boldsymbol{\theta}_2 \;+\; \alpha\,\gamma\,G_2\,\Big(\boldsymbol{x}(S_2,A_2) - \sum_b \pi(b|S_2,\boldsymbol{\theta}_2)\;\boldsymbol{x}(S_2,b)\Big)\\
&= [1.75\cdot 10^{-3}, -1.75\cdot 10^{-3}] \;+\; 10^{-4}\cdot 1\cdot 19\cdot \Big([0,1] - \big(0.499\;[0,1] + 0.501\;[1,0]\big)\Big)\\
&= [1.75\cdot 10^{-3}, -1.75\cdot 10^{-3}] + 10^{-4}\cdot 19\cdot [-0.501, 0.501]\\
&\approx [0.80\cdot 10^{-3}, -0.80\cdot 10^{-3}].
\end{aligned}$$

[*What actions is our policy currently showing preference to? Do you find this surprising?*]

Of course we could have used a different set of features for this problem. One that would make use of the information provided us.

If we had more trajectories/episodes, we would go through the same procedure with all of them, continuing with the θ from the end of the previous episode.

[*However, only using the 3 concentrations as provided in the table would not lead to a useful update. Can you tell why? What could you add to those features to have a useful representation?*]

Problem 2 - Discussion

Part a

Considering a Reinforcement Learning algorithm in general, what is the overall effect of increasing the learning rate? What happens when you set it too high? What happens when you set it too low?

Answer:

The learning rate η is usually not critical for simple deterministic problems, and relatively large values are often useful to reduce the time to convergence. However, if the initial TD-type errors are large compared to the relevant range of values, then it may be necessary to limit the learning rate from the beginning. If the rewards (or actions) are stochastic, then it is necessary to reduce the learning rate according to the Robbins-Monro conditions (see lecture 2, slide 12), although in practice a quicker decay of the learning rate is usually preferable, i.e. the learning rate may need to be decreased so that it reaches zero in finite time. See also momentum techniques (for example Sarigül and Avci [2018]). Also note that for RL methods based on function approximation, the Robbins-Monro conditions are not sufficient for convergence.

Difficulties can arise, when several learning rates (or the learning rate and the exploration rate) interact, e.g. if a sliding average with its own learning rate enters the updates, if a training algorithm is used to approximate the policy or value function etc. In this cases, some consideration of the relative "speed" of the respective learning processes as well as some experimentation may be necessary.

Part b

Is the discount factor γ:

1. Part of the definition of a Markov Decision Process? That is, a part of the definition of the problem to be solved; or
2. Is it external to the problem? That is, a hyperparameter for training the model.

When a discount factor is close to 1, we end up with a long horizon problem. That is, we plan for long-term gains. Assume we were training a Reinforcement Learning agent for a long-horizon problem. Could you think of a reason for which a method using short-horizon targets (cut-off at some horizon h) might outperform a method using long-term horizon targets on this problem? To aid in answering, consider searching online for "planning horizon reinforcement learning model accuracy" and look for related work.

Answer:

The answer is, arguably, both. The discount factor is occasionally included in the tuple defining Markov Decision Processes (MDP) Silver [accessed 2020], but is also frequently omitted and is not present in the original definition Bellman [1957]. That being said, the discount factor determines the relative importance of rewards, based on how close in time they are received. This affects what the optimal policy for the problem would be. If we then were to adopt the view that the discount factor is not part of the definition of an MDP, then an MDP would not in-and-of-itself completely define our control problem (we would not have fully defined our cost function).

There is work indicating that we can control the discount factor during training in a way that would improve the learnt policy, in terms of performance when deployed Jiang et al. [2015]. Specifically, this work defines a new "evaluation" discount factor γ_{eval}, which is smaller than the discount factor we have defined for our problem γ). This is then used to train a policy for a long horizon problem (as defined by γ) using short horizon samples (as defined by γ_{eval}). This means that the discount factor is used here as a training hyperparameter to trade off bias and variance (see also e.g. Schulman et al. [2015]). In particular, the authors show that you can see increasing the discount factor (and therefore the effective horizon) as increasing the complexity of the learnt model. And as you will recall from Machine Learning, we need to match the complexity of our model to the task at hand.

Part c

What other parameters are relevant in Reinforcement Learning, and how do you set their values?

Answer:

Different algorithms have different parameters, so, if you read or hear about a new algorithm, try to get an idea whether their particular parameters are sensitive, need online adjustment, or depend strongly on the problem. In other words, own experience in using an algorithm is useful. Here, we restrict ourselves to the following parameters:

- The exploration rate ϵ has been discussed in the context of MAB and many of the features observed there, carry over also the general case, although this parameter needs to be checked in any particular case and to be adapted in many cases.
- Resolution parameters determine how many discretisation steps are made available for the description of the state of a problem. Likewise, dimensioning parameters decide the complexity of function approximation methods, e.g. the number of neuron is a network or the number of samples. Given the problem complexity, these parameters are determined by the required efficiency of the algorithm and the available resources.
- Initialisation values can include any prior knowledge on the problem, but should otherwise not introduce an unwanted bias. E.g., optimistic initialisation, if done in a suitable way, introduces a bias towards exploration that is often desirable. In other cases, small random initial values are a good choice.
- The number of time steps should be large enough to reach a certain quality of a solution, i.e., so that given the exploration rate, the learning rate(s) and the complexity of the problem the solution can actually be found. It is often helpful to make a rough calculation to get an idea how long you'll have to wait for a solution or whether you should reconsider the choice of the parameters.

References

P. Andreadis. Reinforcement Learning Tutorial 5, Week 6 — Reward Shaping / Semi Gradient Monte Carlo. https://www.learn.ed.ac.uk/webapps/blackboard/execute/content/file?cmd=view&mode=designer&content_id=_5795113_1&course_id=_82627_1, 2021.

Richard Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957. ISSN 00959057, 19435274. URL http://www.jstor.org/stable/24900506.

Darius Braziunas. Computational approaches to preference elicitation. *Department of Computer Science, University of Toronto, Tech. Rep*, page 62, 2006.

Nan Jiang, Alex Kulesza, Satinder Singh, and Richard Lewis. The dependence of effective planning horizon on model accuracy. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, page 1181–1189, Richland, SC, 2015. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450334136.

Mehmet Sarigül and Mutlu Avci. Performance comparison of different momentum techniques on deep reinforcement learning. *Journal of Information and Telecommunication*, 2(2):205–216, 2018.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

David Silver. Applications of reinforcement learning in real world. https://www.davidsilver.uk/wp-content/uploads/2020/03/MDP.pdf, accessed 2020.