Lab Tips

N-Step Q-Learning

$$Q(s,a) \leftarrow Q(s,a) + \alpha[\underbrace{R(s) + \gamma \max_{a'} Q(s',a')}_{\text{target}} - Q(s,a)]$$

- The Q value target can be very bad at the beginning of training!
- This is especially true if rewards are delayed!

Delayed rewards



In Atlantis, firing a gun does not give immediate reward.

- If we hit a ship we only find out several frames later

$$\text{Target: } R(s) + \gamma \max_{a'} Q(s', a')$$

For Q(s, fire):

0 reward!

Poor estimate!

Can we make sure Q(s, fire) sees it's reward directly?

N-Step Q-Learning

Idea: Track reward over a few steps before using estimate

Recall that Q value approximates future rewards:

$$\mathrm{R(s)} + \gamma \max_{a'} \mathrm{Q(s',a')} \approx \mathrm{R(s)} + \underbrace{\sum_{t=1}^{\infty} \gamma^t R(s_t)}_{\text{Future rewards}}$$

N-Step target expansion:

$$\mathrm{Target} = \mathrm{R(s)} + \gamma^1 R(s_1) + \gamma^2 R(s_2) + \gamma^3 R(s_3) + \cdots + \gamma^n \max_{at} \mathrm{Q(s_t,a_t)}$$

N-Step Q-Learning

Reward over next n steps

$$Q(s,a) \leftarrow Q(s,a) + \alpha[\underbrace{R(s) + \sum_{t=1}^{n-1} \gamma^t R(s_t) + \gamma^n \max_a Q(s_n,a_n)}_{\text{target}} - Q(s,a)]$$

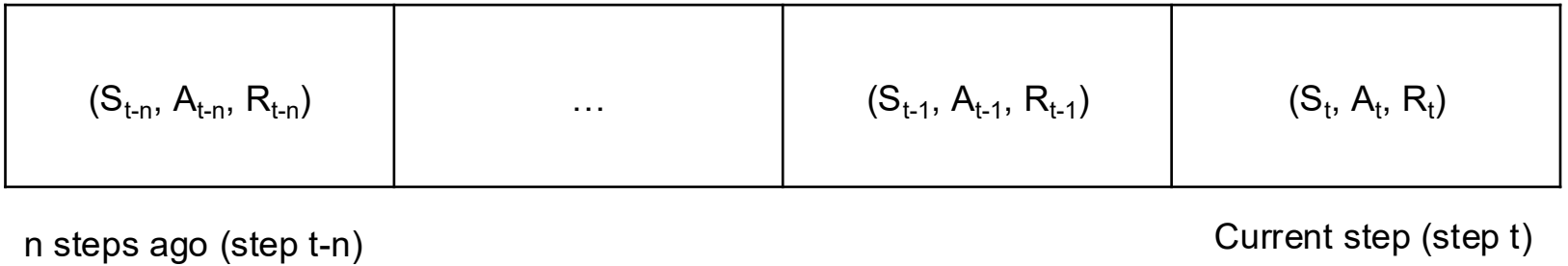
For Atlantis, set n large enough so that the hit registers

*Note we're seeing **one possible** set of n rewards. We might get different results for the same state-action pair in the future!*

N-Step Q-Learning: Implementation

Easiest approach is to change how we add experiences to memory:

- Create a *buffer* to store the last **n** state-action-reward tuples during an episode



- At each step, add the new state, action, reward to the buffer and remove the oldest entry (unless the buffer isn't full yet)
- Update the Q-value for S_{t-n}, A_{t-n}: $Q(S_{t-n}, A_{t-n})$

N-Step Q-Learning: Implementation

Easiest approach is to change how we add experiences to memory:

- Create a *buffer* to store the last **n** state-action-reward tuples during an episode



1-step memory entry

Store (S_t, A_t, R_t, S_{t+1})

N-step memory entry

Store $(S_{t-n}, A_{t-n}, R_{t-n} + \ldots + \gamma^n R_t, S_{t+1})$

Loop through buffer to compute discounted rewards

Note: Also need to change the Q update in replay to discount the estimate for the next state by γ^n instead of γ

N-Step Q-Learning: Tips

- 12-15 are reasonable values for n in Atlantis
- Don’t start updating until the buffer is full!
- You might also think about what to do at the end of an episode, but this isn’t too important

Other lab tips:

- Epsilon should decay slowly, but you might want to test a fully greedy strategy every few episodes to check your learning progress
- You can potentially make the convolutional network faster by introducing strides/downsampling
- Consider changing the random action function, if our random agent fires too frequently, it’s hard to learn! Try making it fire once per second or so.
- If you can’t reliably beat random, document what you tried and what effect it had!