What I have now:

1. All policy gradient method data

Probably I need 2D plots for this one. Will do this on my laptop tonight.

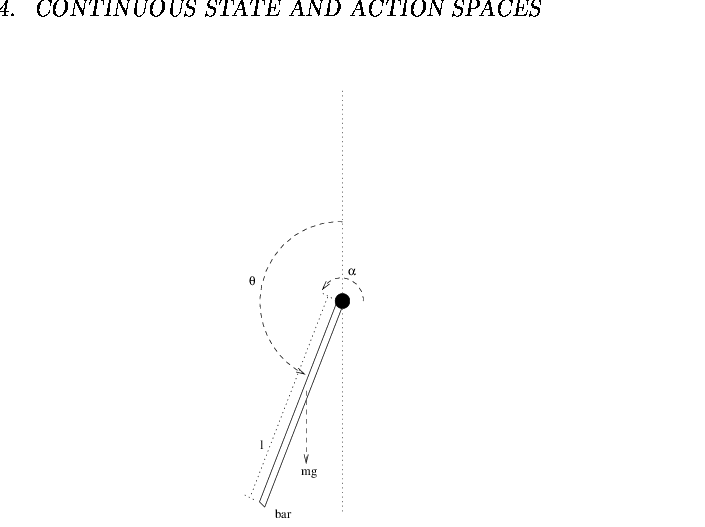
What I need now:

1. For 20000 runs, average reward step size 2^-10, 2^-8, 2^-7…2^-2 2D plots.
2. For 50000 runs…
3. For 100000 runs…. Start running tonight, will get the result tomorrow afternoon.
4. 15000 could takes too long, I will run it if time permits… (This and steps above a trying to make something about the relationship between number of runs and convergence.
5. Run an extremely long one to say it won’t really work.
6. Comparison between different number of tilings. For example, 32, 64, 128, tilings for 20000 runs. (With optimal parameters)
7. Use some large number of tilings to redo 2D plots
8. Find the most optimal one
9. Do the same for sarsa with eligibility traces.

Environment & problem

The problem that we are interested in is the Pendulum Swing Up problem, as mentioned in <http://www.incompleteideas.net/papers/SSR-98.pdf> at page 33.

Since details about the environment can be found in the paper, we would only make a brief description about the problem here.



<https://www.researchgate.net/figure/The-Pendulum-Swing-Up-Problem-A-bar-hanging-from-one-extremum-and-subject-to-gravity-and_fig6_2259017>

As shown in the figure above, we have a pendulum (or say the bar in the figure) hanging in the space with one end attached to a fixed point. The pendulum (bar) is subjected to gravity (mg) and some torque which can give a angular acceleration of the pendulum(bar) and swing it up.

Our goal in the problem would be first applying a sequence of torque on the pendulum to swing it straight up and then keeps applying torque to hold it stationary against the gravity, and some potential horizontal turbulence (i.e. a horizontal wind).

To better demonstrate the outcome of our reinforcement learning algorithms and their comparison on convergence speed and final error, we limited the max torque (acceleration) that can be applied to the bar such that the agent needs multiple swing in alternating direction to swing it up. To be more specific, ….

Also, to simplify the problem, we only allow discrete action space for the following experiment. To be more specific, we allow 3 possible actions: apply positive torque (accelerate clockwise), apply negative torque (accelerate inverse clockwise) and apply zero torque (do nothing).

To reduce the complexity, we simply set the reward signal at every time step to be negative absolute angle between the bar and the vertical straight line.

Note: this problem is naturally a continuing problem.

Algorithm to solve the problem

We are planning the try multiple algorithms on the problem, including Sarsa, Sarsa(lambda) and Actor-Critic. Note: all algorithm mention above will be used with function approximation.

(we were planning to try off-policy algorithm as well but due to time limit we failed to do so).

Sarsa

Actor-Critic algorithm (a policy gradient method)

Instead of deriving the policy from value functions (i.e. action values), policy gradient is a method to directly optimize policies. To be more specific, policy gradient method first calculates the approximation of the gradient of some evaluation function with respect to policy parameters, then travel along the gradient to gain a better policy. By iteratively doing so, the policy eventually converges to the optimal policy.

Experiment: solve the problem with our algorithms

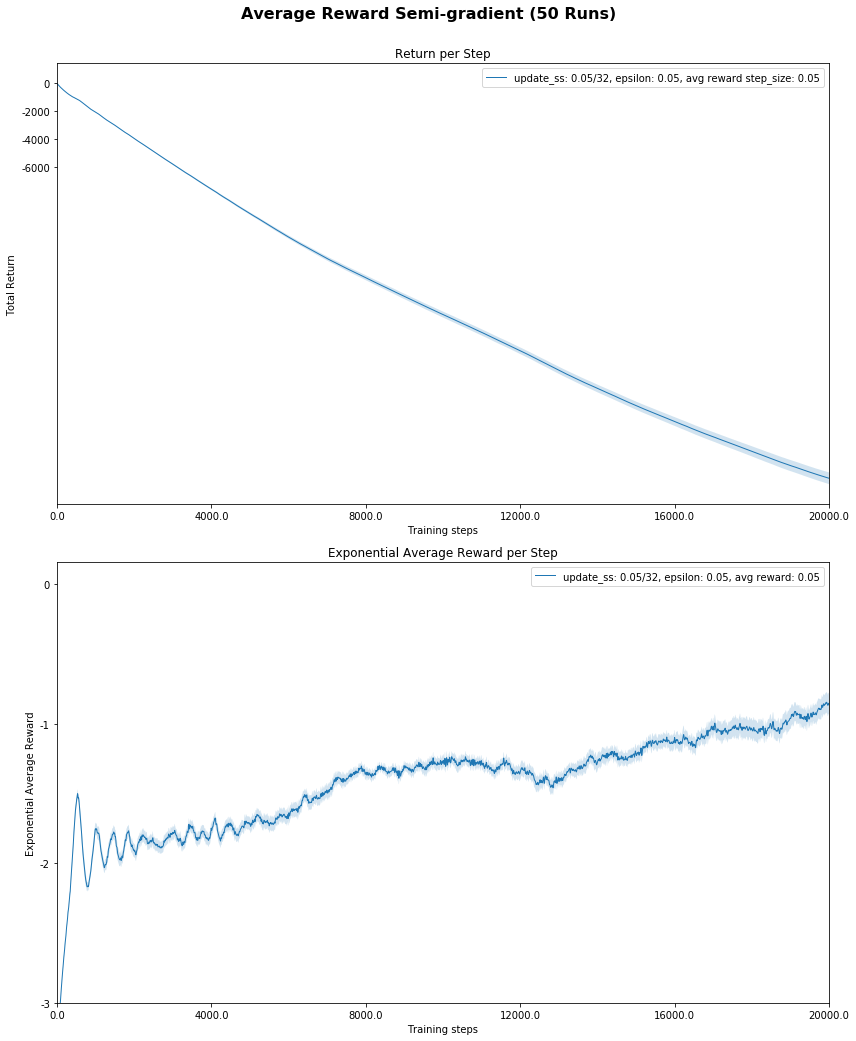
In the experiment with policy gradient,

Questions…

Why are we always using one step algorithm here? What would be the comparison between it and n-step methods?

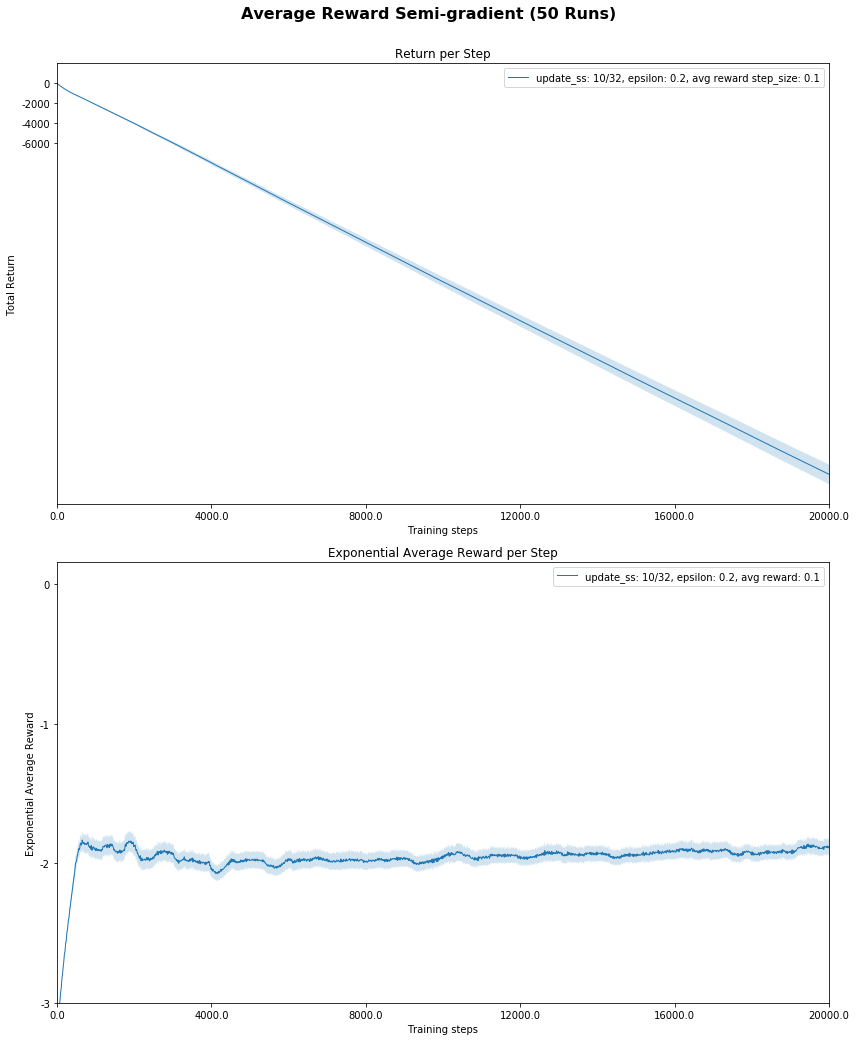
Do we need to subtract the baseline from the sarsa differential return?

How did we select the step-size parameters? Chapter 9.6



This is the best parameter setting for 20000 runs with 32 tilings . It cannot converge to a good policy due to limited time steps.

As we know, increasing step-size and epsilon can allow faster learning, but as shown in the following figure, large step size and large epsilon can only converge to a bad policy



Relationship between epsilon and degree of discrimination in function approximation (number of tilings)

How do I verify whether a policy gradient method converged to a deterministic policy or random policy?

A problem worth thinking is, what’s the state distribution in each receptive field when we are using different level of exploration?

Assuming we have a near optimal deterministic policy, the policy should always take the same action on states within each receptive field and maps to some other receptive field. Since the policy is fixed, in this continuing case, state distribution is fixed.

Without exploration, certain state in each receptive field will witness a high frequency and the policy is made based on them. (that’s indeed how the deterministic policy worked, the deterministic policy is made to maximize the utility of the dominant states in each receptive field).

However, with constant exploration, say the constant epsilon we involved in our sarsa agent, things become different.

When we have high discrimination function approximation, for an arbitrary state in each receptive field, the policy almost certainly maps to another receptive field, especially in this kind of simple stationary physics environment. Therefore, even if the exploration steps takes up to some bad situation (i.e. a bad receptive field), we have relative confident guess about it’s explicit position in that bad receptive field and can make a corresponding decision to take him out.

And vice versa, with low discrimination function approximation, when the exploration steps takes us to some receptive field, it is very possible for it to hit a low-frequency state and therefore the policy cannot make the right decision and cannot take it out. This is especially common for relative large epsilon.

This is actually something I found during the parameter testing of sarsa… I initially don’t understand why there would be a relationship between function approximation and epsilon in terms of the final average reward. After making the plot of the final average reward (after a long long time steps) for a epsilon with different number of tilings, I realized this.