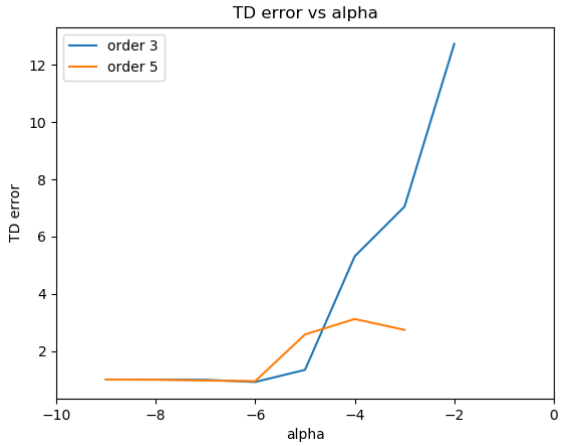

Figure 1: cartpole td error
TD error goes to nan (very high value) for $alpha >= 0.01$ for 5^{th} order
TD error goes to nan for $alpha >= 0.1$ for 3^{rd} order

CMPSCI 687 Homework 3

Due November 5, 2018, 11:55pm Eastern Time

Instructions: This homework assignment consists of a written portion and a programming portion. Collaboration is not allowed on any part of this assignment. Submissions must be typed (hand written and scanned submissions will not be accepted). You must use LaTeX. The assignment should be submitted on Moodle as a .zip (.gz, .tar.gz, etc.) file containing your answers in a .pdf file and a folder with your source code. Include with your source code instructions for how to run your code. You may not use any reinforcement learning or machine learning specific libraries in your code (you may use libraries like C++ Eigen and numpy though). If you are unsure whether you can use a library, ask on Piazza. If you submit by October 14, you will not lose any credit. The automated system will not accept assignments after 11:55pm on October 14.

:

- Solution 1:

  Before implementing the methods, I was under the impression that $v(s)$ always converges in mean to v^{π} for a tabular representation when step size is sufficiently small and it converges to v^{π} in the case of linear function approximator provided $v(s)$ can be represented by $\phi(s)$ and step size is sufficiently small.

  I missed out on the standard assumptions of convergence like $\sum \alpha = \infty$ and $\sum \alpha^2 = 1$ and decaying step size and came across them when I was

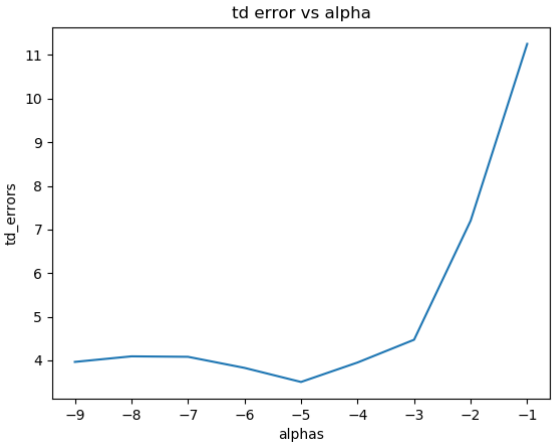

Figure 2: gridworld td error

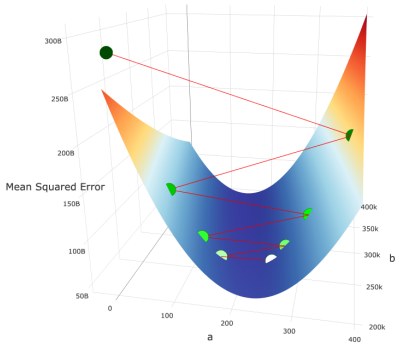

Figure 3: Exploding gradients

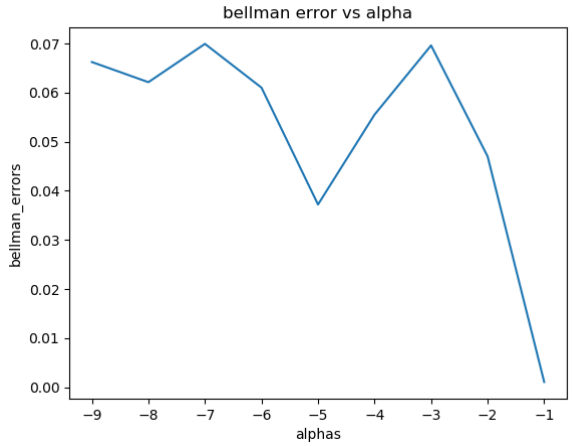

Figure 4: Mean squared bellman error for gridworld

trying to find out the reasons for TD diverging. I did not expect TD error to diverge but it diverged for higher values of step size ie. This was probably because learning rate ie- step size in the TD update was too high (gradient descent) .

In a normal gradient descent, if your learning rate is too high, it oversteps the minima and oscillates around it.

Particular for the case of divergence what happens is that as soon as an oversized step is taken from an initial point $p(i = 0)$, the gradient descent algorithm lands to a point $p(i = 1)$ that is worse than $p(i = 0)$ in terms of cost. At this new but cost function-wise worse point $p(i = 1)$, when recalculating the gradients, the gradient values are increased, so the next (hopefully corrective) step is even larger. Nevertheless if this next step leads to a point $p(i = 2)$ with even larger error because we overshoot again, we can be led to use even larger gradient values, leading ultimately to a vicious cycle of ever increasing gradient valuesand "exploding coefficients" .This can be visualized through figure 3.

I expected it to converge for lower learning rates below 10^{-3} since the target is moving and I found that it started converging at $stepsize = 10^{-5}$ for grid world and $stepsize = 10^{-5}$ for cart pole 3rd order and 10^{-6} for 5th order.

The step size for convergence for order 5 was expected to be lower than that for order 3 since td depends on v^{π} which is a product of weights and Fourier basis of order n .

Since the number of parameters in Fourier bases order 5 is larger than Fourier bases of order 3, v(s) would have larger variance when represented using Fourier bases of order 5

(increased model complexity always decreases bias and increases variance) and hence require smaller step size or learning rate.

Also, 100 episodes is too low for estimating the value of $v(s)$. I was expecting $v(s)$ to be around 40 it was mostly random small numbers for lower values of alpha as it was hardly getting updated and in the range of $[30, 60]$ for high value of alphas.

In this case, since the policy followed is a random policy (stochastic policy), when v is equal to v^{π} MSTDE may not go to zero but mean squared bellman error should go to zero. Figure 4 shows the mean squared belllman error for different values of alpha.

If it was following a deterministic policy, MSTDE would have been zero when $v = v^{\pi}$ after sufficiently large no of training episodes. (1)

- To plot the graphs for gridworld run $->$ python *run_policy.py*

- To plot the graphs for cartpole run $->$ python cartpole.py