Itai Nuri 041704990

Guy Gabay 205685928

Reinforcement learning HW2

Section 1 – Monte-Carlo Policy Gradient (REINFORCE) (50%)

- What does the value of the advantage estimate reflect?

The value of the advantage reflects a fix that should be applied to the value function at the current state to make it more correct according to the current policy. When we apply a small change to the value function in the direction of the gradient (gradient descent/ascent) we make its output closer to the real reward expected according to the current state and policy, we use a baseline (get the value function closer to the baseline), that means that the baseline needs to reflect the true rewards and so needs to be unbiased by the actions. It can be seen as how much better or worse we were at assessing the reward.

Why is it better to follow the gradient computed with the advantage estimate instead of just the return itself? (5%)

when we train, we use sampling of actions according to probabilities or relative rewards. So the important point is the ratio between rewards of different paths and not an individual reward of this or that action, by comparing to a baseline we make the gradients smaller and emphasize the differences between rewards and make the variance of the rewards smaller which reduces noise and convergence.

- The reduction of the baseline **does not** introduce bias to the expectation of the gradient since: $\mathrm{E}_{\pi_\theta}[\nabla log\pi_\theta(a_t|s_t)b(s_t)] = 0$ What is the prerequisite condition for that equation to be true? Prove the equation (10%)

The perquisite is that the expectation of the baseline $b(s_t)$ is 0 and that it isn't dependent on the action:

$$\mathrm{E}_{\pi_\theta}[\nabla log\pi_\theta(a_t|s_t)b(s_t)] =$$

$$baslene\ doesnt\ depend\ on\ parameters\ \theta = \mathrm{E}_{\pi_\theta}[b(s_t)\nabla_\theta log\pi_\theta(a_t|s_t)] = smoothing\ theorem$$

$$= \mathrm{E}_{\pi_\theta}\left[b(s_t)E[\nabla_\theta log\pi_\theta(a_t \mid s_t)]\right] = derivation\ of\ the\ log = \mathrm{E}_{\pi_\theta}\left[b(s_t)\sum \nabla_\theta\pi_\theta(a_t|s_t)\right]$$

$$= liniarity = \mathrm{E}_{\pi_\theta}\left[b(s_t)\nabla_\theta\sum \pi_\theta(a_t|s_t)\right]$$

$$= sum\ of\ all\ probabilities = \mathrm{E}_{\pi_\theta}[b(s_t)\nabla_\theta 1] = 0$$

Therefore, we need the baseline to be independent of the actions-policy and to be unbiased-have 0 expectation.

Scripts general notion

- For the script parts we changed the original file and added 2 flags for the 3 modes (explained at the end).
- For the actor critic and baseline we added another network – ValueNetwork, the same for both.
- The PolicyNetwork has a different class for each part (PolicyNetwork, PolicyNetwork_bl, PolicyNetwork_ac) but the NN architecture is the same for all 3 modes.
- We left the PolicyNetwork unchanged and for the ValueNetwork we used 3 FC layers with biases of widths `[state_size, 64],[64, 16], [16, 1]`.

Baseline script

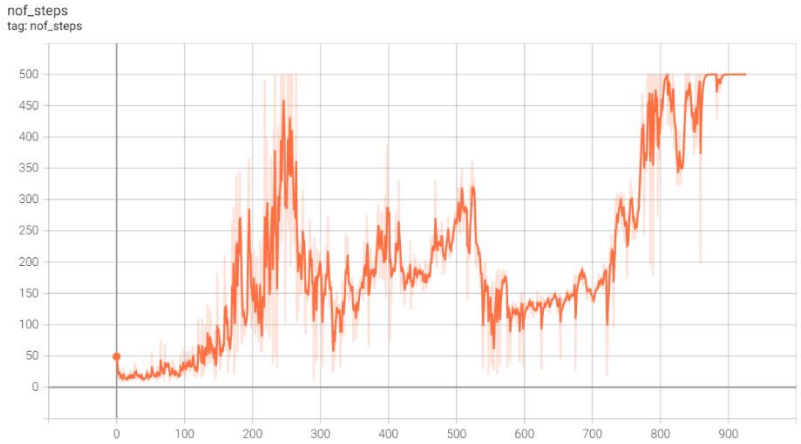
a) Starting with the original network

we set the learning rate to be:

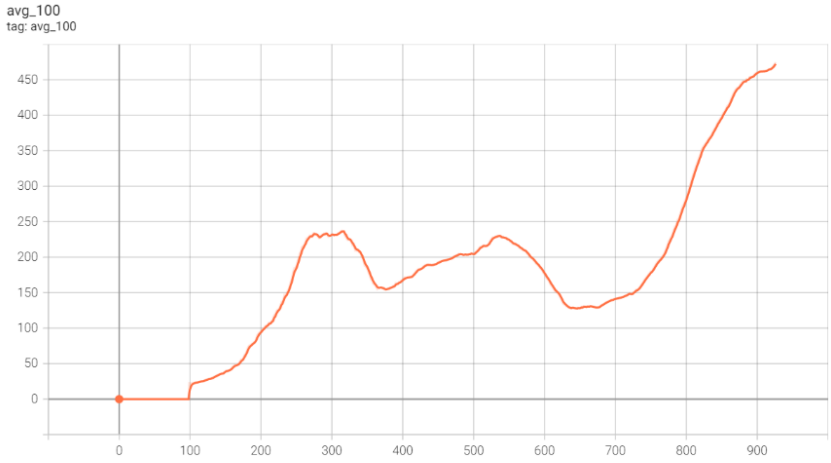
PolicyNetwork - 0.0005

we got an average of 475.87 after 927 iterations and 533 seconds(`Solved at episode: 927 total time: 533.5228073596954`)
also (`Solved at episode: 706 total time: 320.06550574302673`).
also(`Solved at episode: 545 total time: 202.10132241249084`)
also(`Solved at episode: 1782 total time: 1109.8261880874634`)

number of iterations until done vs episode



number of iterations until done averaged over 100 episodes vs episode



b) on the baseline mode

we set the learning rate to be:

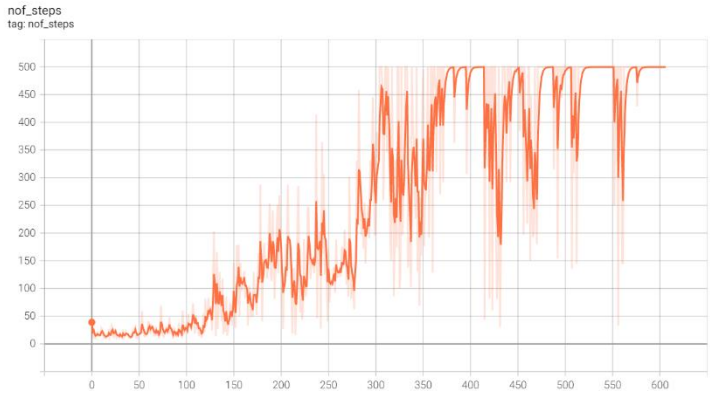
PolicyNetwork - 0.0006

ValueNetwork - 0.004

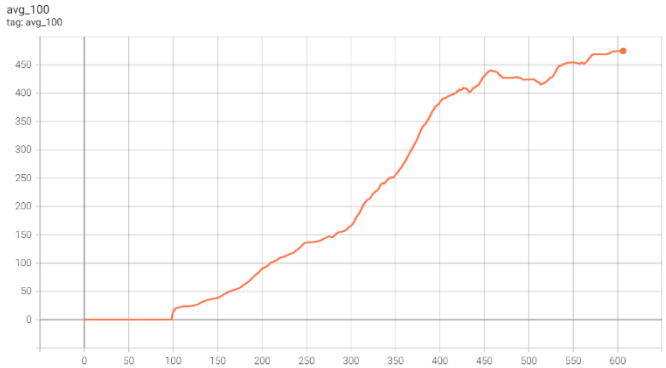
we got an average of 478 after 607 iterations and 559 seconds (`Solved at episode: 607 total time: 559.6108968257904`)

also (`Solved at episode: 725 total time: 473.5974485874176`)

number of iterations until done vs episode



number of iterations until done averaged over 100 episodes vs episode



- Compare the results before and after the change (please refer also to convergence time).
  - When we look at the graphs we can see that the baseline converges after fewer iterations ~600 instead of ~900. There were many trials and in general there is high variance on the results, and with the best configurations the baseline preformed slightly better.
  - Here we can see that both took about the same time (the baseline has an additional network which is also bigger). But on average the baseline was faster to converge and more importantly much more stable between different runs.
  - If we will examine the number of steps until done graphs, we will notice that the baseline graph is much more stable between episodes, which corresponds to the explanation above.
  - Also, when training we noticed that the regular version is much more sensitive to changes on the learning rate and much less forgiving in a way that a small change in the learning rate can greatly deteriorate the convergence, where on the baseline its less sensitive.

Section 2 – Advantage Actor-Critic (50%)

- Why is using the TD-error of the value function for the update of the policy network parameters is practically the same as using the advantage estimate (of equation 1)? Prove. (5%)

  The value of a step depends on the rewards of all following steps, the TD error is the difference between the current estimate for the return value of current time t, the discounted value estimate of the return value of next time t+1 and the actual reward gained from transitioning to next state, The advantage estimates the difference between the overall reward estimation and the reward from next state – Q function so the same as r+v(s').

  ~~so the TD error is used to correct the current assumed state value according to (in the direction of) an updated next step value.~~

  Proof:

$$TD_e = r + \gamma V(s') - V(s)$$

$$\widehat{A_t} = \widehat{Q_w}(s_t, a_t) - \hat{V}(s)$$

$$belman\ eqn: \quad Q(s_t, a_t) = E[r_t + \gamma V(s')]$$

$$\widehat{A_t} = r + \gamma V(s') - \hat{V}(s) = TD_e$$

- Explain which function is the *actor* and which is the *critic* of the model and what is the role of each one. (3%)

The policy means to choose an action so is the actor, and wants to learn the best action for every scenario - or a policy, but best action according to what? According to the value of the new state, therefore the value function is the critic which "grades" the actions (indirectly) done by the actor. So eventually the actor choses an action according to its policy, the critic watches the rewards according to their definition and updates the value of the state and the actor to change its policy according to this value.

Actor critic script

we set the learning rate to be (same as baseline):

we set the learning rate to be:

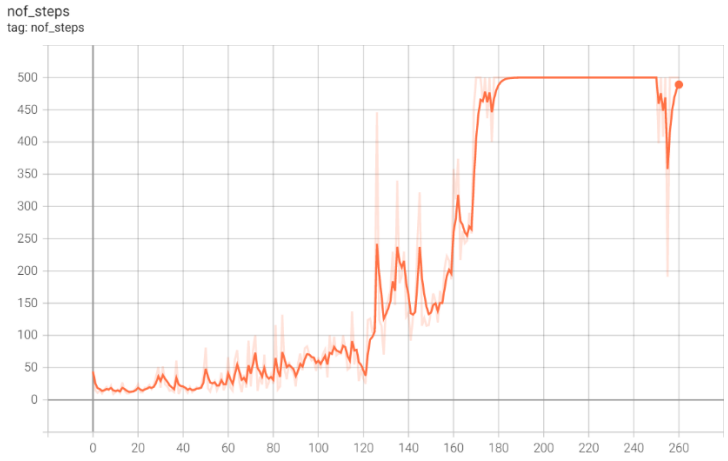
PolicyNetwork - 0.0006

ValueNetwork - 0.004

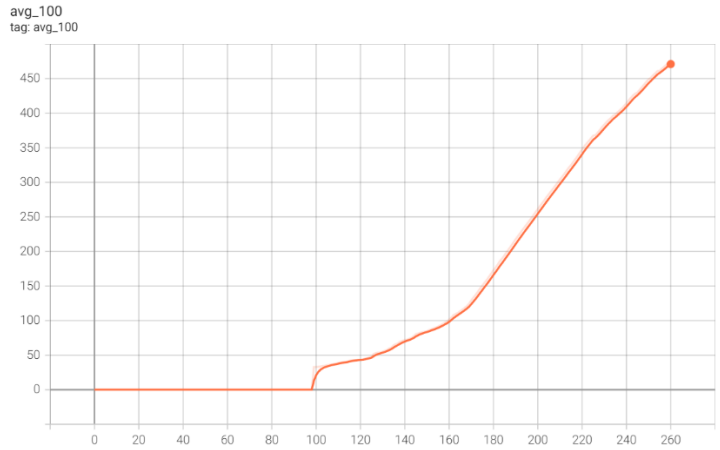
we got an average of 476.38 after 261 iterations and 215 seconds (`Solved at episode: 261 total time: 215.00902104377747`)

another (`Solved at episode: 246 total time: 288.8289213180542`)

number of iterations until done vs episode



number of iterations until done averaged over 100 episodes vs episode



- We chose to use the same ValueNetwork NN layers so that we will be able to compare the 3 architectures.
- We tried different learning rates and got the best results with the same learning rates as the baseline case.
- We can see that the actor critic converges much faster than both other architectures in both time and number of episodes.
- Looking at the graphs we can see that is also very stable.
- It is also very forgiving for learning rates changes, could be because there are many small changes that average rather than full paths which can stray off and impact more dramatically.
- We tried many times and got similar results.

Instructions

there is 1 file that runs all 3 models, flags:

- Mode flags
  do_baseline - run baseline model
  do_actor_critic - run actor critic model
  - if both flags are 0 – runs the original model
  - cant both be 1 at the same time

- learing rates
  learning_rate1 = learning rate for PolicyNetwork
  learning_rate2 = learning rate for ValueNetwork
  - learning rates can be changed at beginning of ***run()*** function (different set for each mode case)

- tensorboard
  - ***tb_dir_str*** = **"tb_dir"** directory for tensorboard file
  - tensorboard --logdir=rein_learning/hw2/tb_dir --bind_all