Problem 1 (a)

$$\sum_{t=1}^{T} \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \left(b(s_t) \right) \right]$$

$$= \sum_{t=1}^{T} \mathbb{E}_{p(a_t,s_t)} [\mathbb{E}_{p(\tau|a_t,s_t)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \left(b(s_t) \right) \right]]$$

Since τ *and* $p(\tau|a_t, s_t)$ *is irrelevant to* θ*, we then need to prove*

$$\sum_{t=1}^{T} \mathbb{E}_{p(a_t,s_t)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \left(b(s_t) \right) \right] = 0$$

Which means to prove $\sum_{t=1}^{T} \mathbb{E}_{p(s_t)} [\mathbb{E}_{p(a_t|s_t)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \left(b(s_t) \right) \right]] = 0$

$$\sum_{t=1}^{T} \mathbb{E}_{p(s_t)} [\mathbb{E}_{p(a_t|s_t)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \left(b(s_t) \right) \right]]$$

$$= \int_{s_t} \int_{a_t} p(s_t) \pi_\theta(a_t|s_t) \nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t) da_t ds_t$$

$$= \int_{s_t} p(s_t) b(s_t) \nabla_\theta \int_{a_t} \pi_\theta(a_t|s_t) da_t ds_t$$

$$= \int_{s_t} p(s_t) b(s_t) * 0 \; ds_t$$

$$= 0$$

Problem 1 (b) (1) According to Markov Chain property, each event depends only on the state attained in the previous event. So conditioning on s_t* is inherently conditioning on (s_{t-1}, a_{t-1}) and therefore all (s_i, a_i) where i < t*. So conditioning on s_t* is equivalent to

conditioning on $(s_1, a_1, ..., s_{t-1}, a_{t-1}, s_t)$.

$$
\begin{aligned}
&\sum_{t=1}^{T} \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \left(b(s_t) \right) \right] \\
&= \sum_{t=1}^{T} \mathbb{E}_{p(s_{1:t}, a_{1:t-1})} [\mathbb{E}_{p(s_{t+1:T}, a_{t:T}|s_{1:t}, a_{1:t-1})} \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \left(b(s_t) \right) \right]] \\
&= \sum_{t=1}^{T} \mathbb{E}_{p(s_t)} [\mathbb{E}_{p(s_{t+1:T}, a_{t:T}|s_t)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \left(b(s_t) \right) \right]] \\
&= \int_{s_t} \int_{a_t} p(s_t) \pi_\theta(s_{t+1:T}, a_{t:T}|s_t) \nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t) da_t ds_t \\
&= \int_{s_t} p(s_t) b(s_t) \nabla_\theta \int_{a_t} \pi_\theta(a_t|s_t) da_t ds_t \\
&= \int_{s_t} p(s_t) b(s_t) * 0 \; ds_t \\
&= 0
\end{aligned}
$$

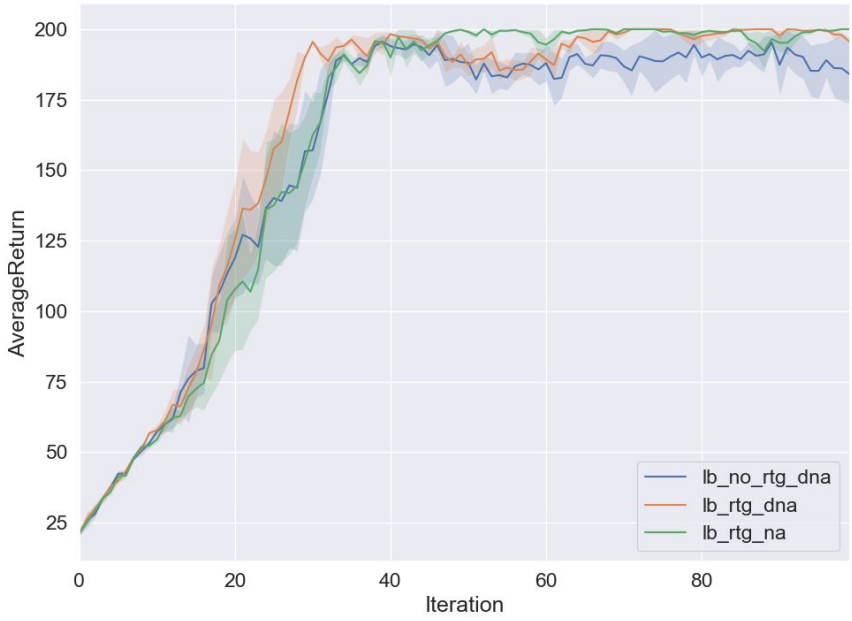
Usage and command line expressions are in README.md

For all my hidden layers, I used xavier initializer.

Problem 4:

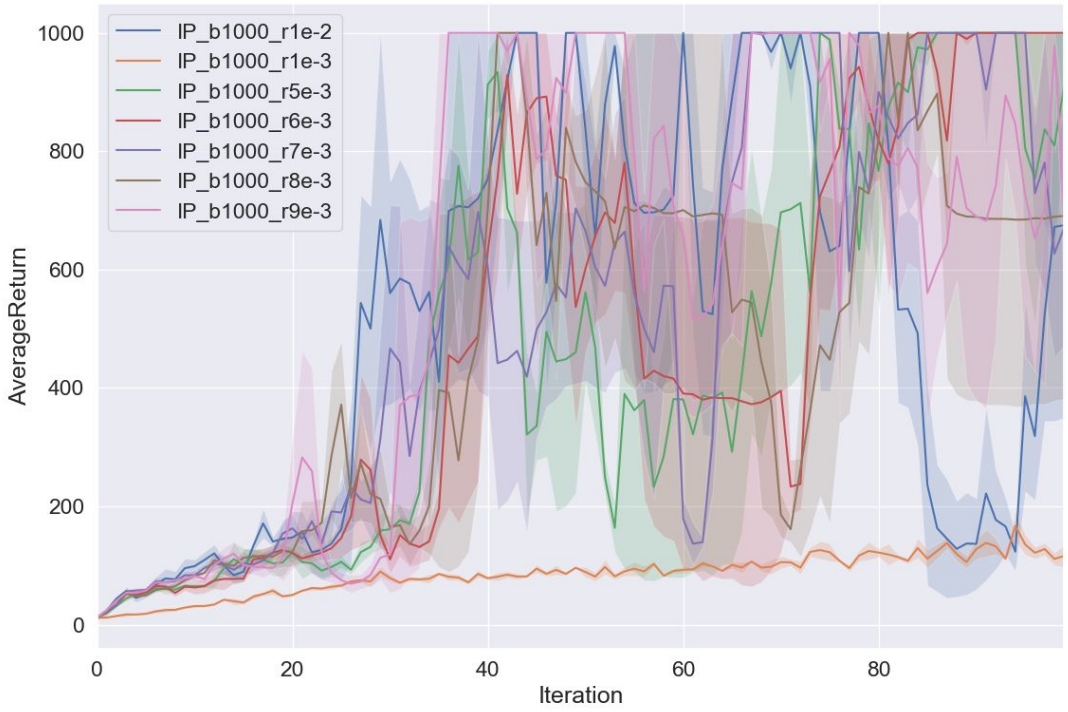
- NN structure:
  - one 32-neuron fully connected hidden layer
  - learning rate: 1e-2
  - max_iteration: 100
  - seed: 3
- Without advantage-centering, reward-to-go has better performance than the trajectory-centric NN.
- Advantage-centering is definitely helping, regardless of whether you are using reward-to-go or not. But surprisingly, advantage centering without reward-to-go seems to have the best performance.

- Larger batch size can help a smoother convergence with fewer iterations.

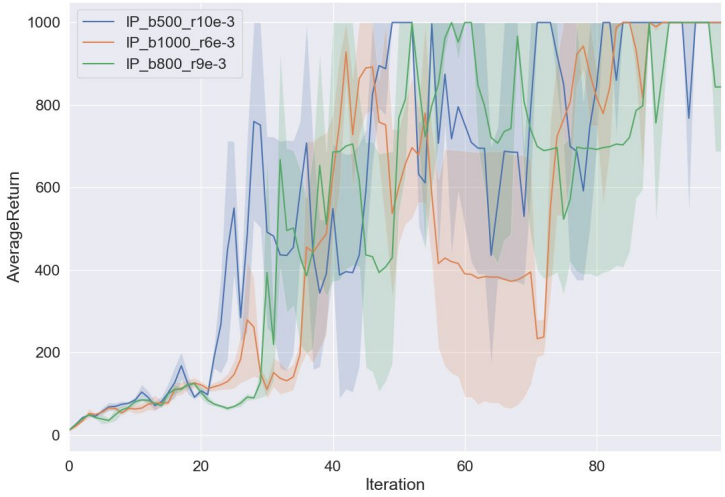


Problem 5:

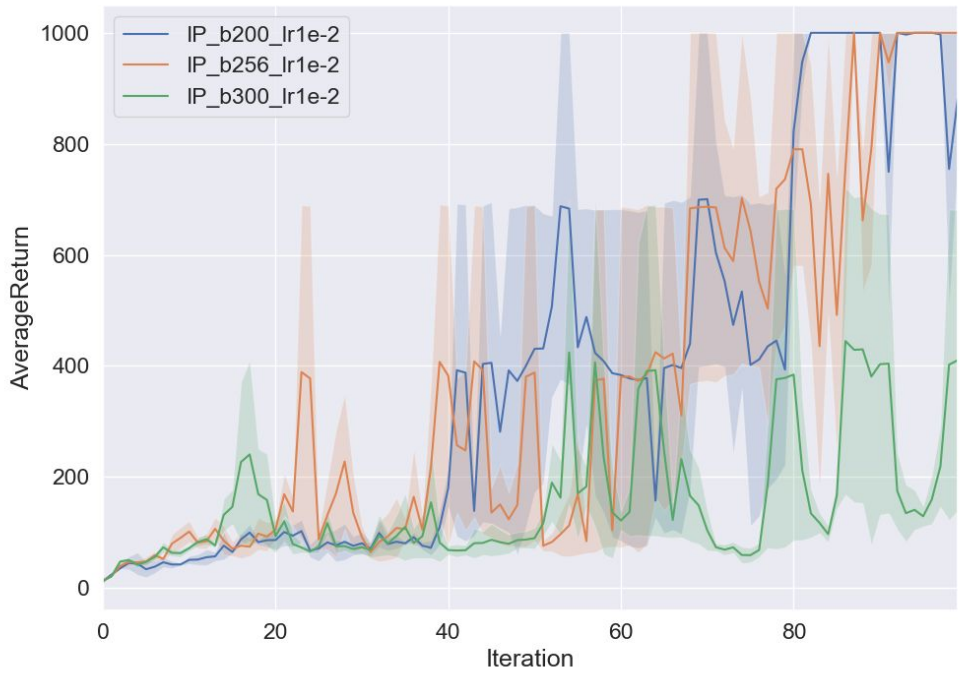
- NN structure:
  - two 64-neuron fully connected layers
  - max_iteration: 100
  - discount factor: 0.9
  - seed: 3.
- **Optimal batch size: 200, optimal learning rate: 1e-2.**
- I first tried batch size 1000 with different learning rates and lr=5e-3/6e-3 turned out to have the best results.



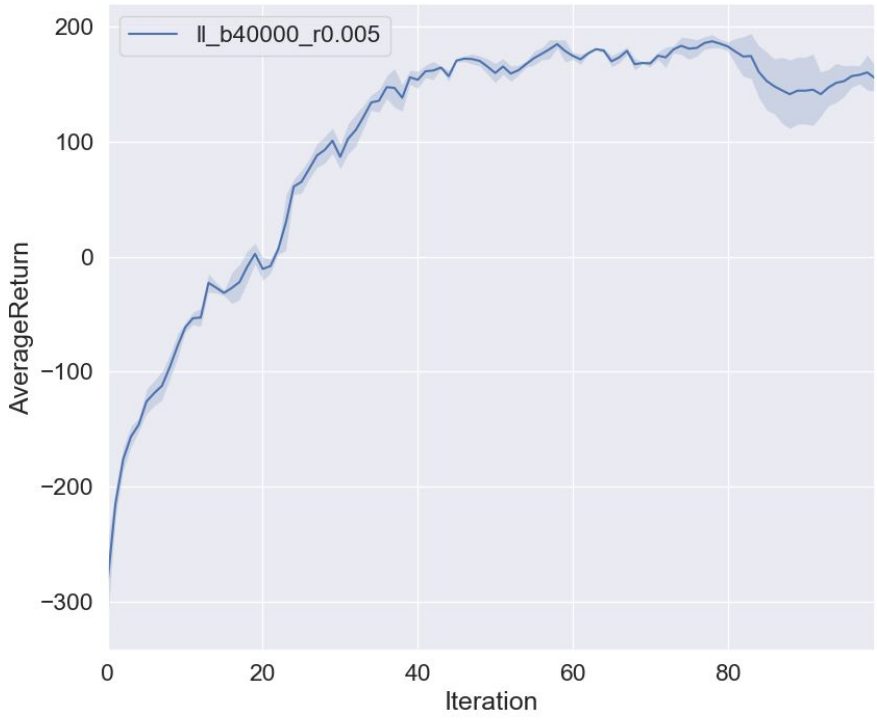
- Then I tried combinations of batch size 500-1000 (incremented by 100) and learning rate 5e-3-1e-2 (incremented by 1e-3)
- After comparing all the graphs, as shown in the graph below, b800_r9e-3, b1000_r6e-3, b500_r1e-2 have the best performances among all configurations. But I think b500_r1e-2 has the smoothest and most stable convergence within 100 iterations (it did still have high variance, but relatively it's the best one I think).



- I also tried batch size 32, 128, 256, only b256_r1e-2 has good performance. After testing batch_size 200 and 300. 200 has the best performance. Optimal batch size: 200, optimal learning rate: 1e-2.



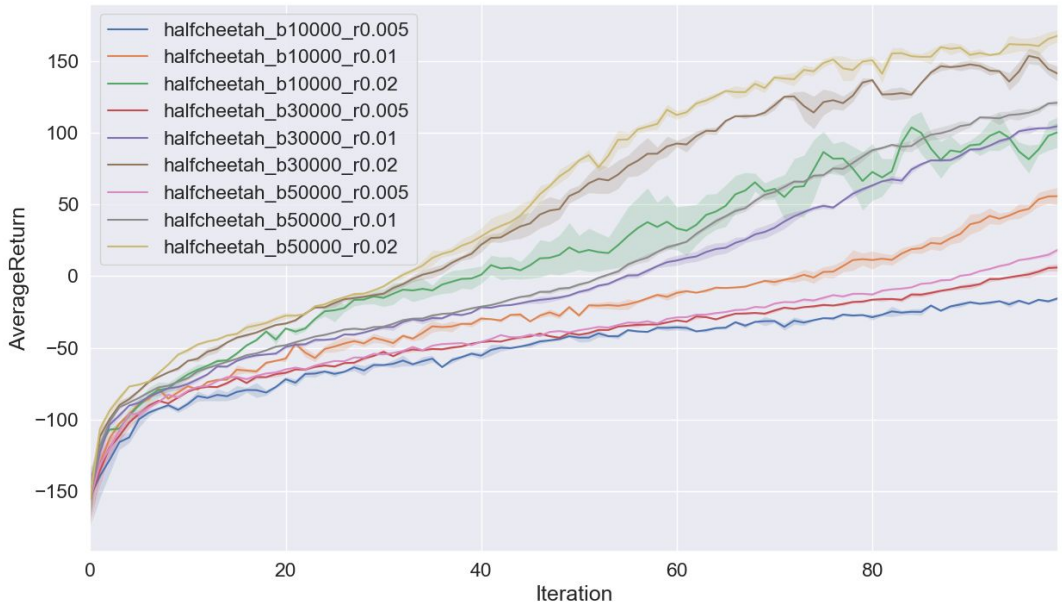
- **Optimal batch size: 200, optimal learning rate: 1e-2.**



Problem 7:

As shown in the graph, it did converged to an average return of around 180. I used the command provided (seed = 3).

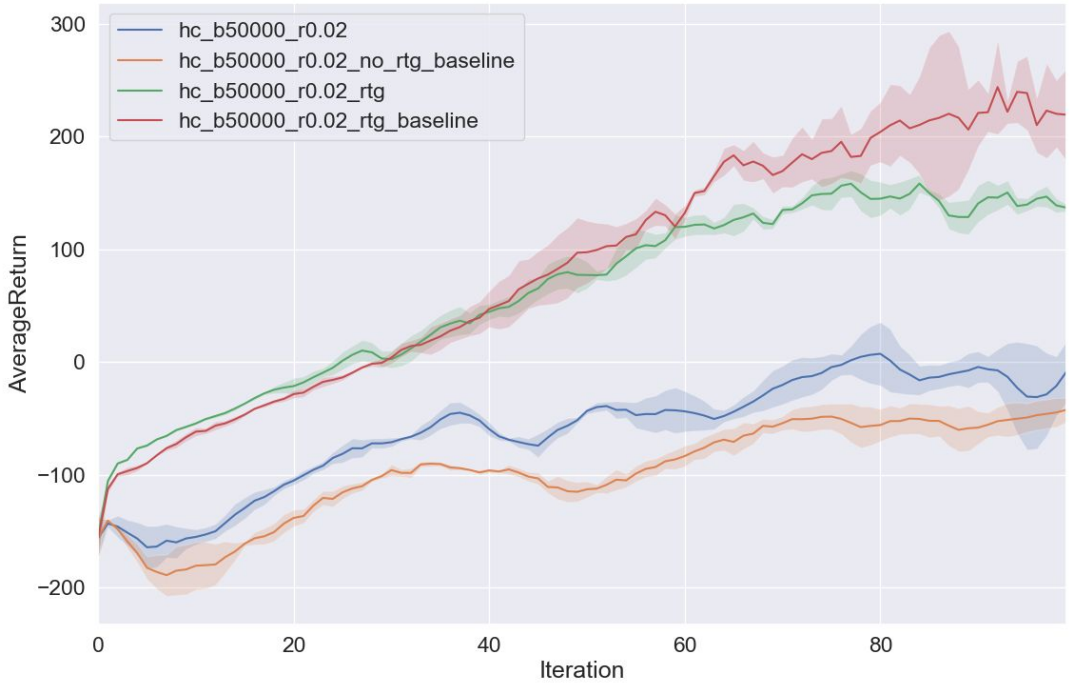
Problem 8:



I think a larger batch size can help smoothen the learning as well as reduce variance. But larger batch sizes did not significantly improve the average returns (If the batch size was incremented by 20000, there was a 20-30 increase for 100 iterations).

However, a larger learning rate significantly improved the average returns while also introduced more variance.

To sum up, I think the best batch_size-learning_rate combinations is 50000 and 2e-2.



With reward-to-go and baseline, it can actually reach an average return of around 200.