Journal Report 4
9/30/19-10/4/19
Pawan Jayakumar
Computer Systems Research Lab
Period 5, White

Daily Log

Monday September 30

I contacted Ofir Nanchum, one of the authors of https://arxiv.org/pdf/1810.01257v2.pdf and asked him whether he considered using a 3 or ¿2 level hierarchy. Few days later he contacted me saying while it is possible to do, it was difficult to do so for him and also that having more levels would yield diminishing returns. He recommended for me to look at https://arxiv.org/abs/1712.00948 which successfully utilized a 3 level hierarchy to form checkpoints for the agent to reach
(https://www.youtube.com/watch?v=DYcVTveeNK0 illustrates this).

Wednesday October 2

I read more about the paper Nanchum linked me and was able to grasp the approach's strengths. The model in the paper (referred to as hierarchical actor critic or HAC) is able to parallely train all layers in the hierarchy at the same time. The reason why this is noteworthy is because the transition tuples for higher levels is dependant on lower levels. For example, if the higher level policy gives sub-goal g1, but the low level policy reached state s1 != g1, how should the 'blame' be distributed. It isnt the upper level policies' fault that the lower level policy didn't reach a goal. At the same time, it isn't the lower levels fault that it hasn't really been optimized yet. In past papers, the levels in the hierarchy had to be trained sequentially. HAC is able to train levels in parallel because the policy at level i assumes the policy at level i-1 is optimal. It is able to do this by incorporating 3 different transition tuples rather than just (state, action , reward, nextState). If I were to explain about these 3 transitions, it would basically mean re-writing the paper, so I'll leave that for the paper to explain.

Friday October 4

I completed the DQN implementation for the cartpole. It wasn't that hard because it isn't a very hard algorithm. The biggest thing was fixing all the syntax errors. In particular, I used batchLen and batchLength interchangeably which messed up the script. After seeing it solve the task for balancing a pole in 600 episodes, I introduced a target network. The idea is that the main network is really likely to over-estimate Q values because it is the one selecting those actions according to a greedy policy (always picking the highest q value option) . With a target network, we can have a less biased method of evaluating q values. The target network wont be updated until after every episode. https://arxiv.org/pdf/1509.06461.pdf explains this in more rigorous terms.

Timeline

Date	Goal	Met
Today minus 2 weeks	Implement a simple ConvNet in tensorflow on the mnist dataset along with generating the tensor graph to view the accuracy over time	Yes I was able to implement the CNN and see the accuracy over time graph
Today minus 1 week	Implement my own CNN that achieves atleast 99 percent on MNIST	Yes it reached 99.4 percent
Today	Finish implementing the DQN for cartpole	yes and was also able to implement DDQN
Today plus 1 week	Adapt the DDQN to play atari breakout	
Today plus 2 weeks	Create an algorithm draft	

Reflection

It was cool to see that my DDQN worked, but it was on cart-pole which isnt that exciting to see. The pole tilts for a second then the agent masterfully balances the pole. Atari breakout is cooler to see work I think. The atari game environment returns an image of the screen as the state so you need to do some preprocessing on each frame and feed it into a cnn.

I am really glad that someone from the papers I read actually responded to my emails. I am going to continue contacting both Nanchum and others who are interested in my project idea.

After each paper I read, I have a better idea of how exactly I'm going to accomplish my project. Considering MuJoCo environments are used in many papers as a benchmark, I am going to use that to validate my results. My goal was to create a new model that can learn to do a variety of tasks. I could let the agent cycle through n different environments after doing k episodes in each one and show that it is eventually able to solve all n tasks faster than n different policies.

I said earlier that I was finished with 1/3 of my model and that I could see the end, but with every paper I read, I don't think that is the case. There is always something newer and better coming out. I think at some point I have to say this is it, but I don't think that time is now. I think until I rigorously define my algorithm, I should continue to read more papers. On that note, I will take a break from implementing models to try and do that because If I can do it in the next 2 weeks, I want to submit it to the Regeneron STS competition. If I don't , that is also fine because I think my project requires a lot more time anyways.