CS285 Final Project

Aryan Luthra

Scholastic Learning

Using Curriculums and Suboptimal Demonstrations to Approach the RL Exploration Problem on Sparse Reward Functions

By Aryan Luthra

Table of Contents

Table of Contents 2

Abstract 3

Background and Introduction 4

Motivation 4

Connection to Human Agents 4

Exploration in RL 5

Methods and Findings 6

Motivation 6

Environment Information and Changes Made 7

Curriculum Development 8

Curriculum Learning Results 9

Incorporating demonstrations 11

Demonstration + Curriculum Results 12

Further Discussions & Applications 13

Discussion of Key Observations 13

Future Works and Applications 15

Conclusions 17

References 18

Abstract

During the average lifespan of a human one spends most of their time “learning”, at least based on the classical RL definition of the term. However, almost all of this learning is done as a transfer of knowledge from one agent to another. This is obviously a good idea because it allows humans to build on top of each other's knowledge as a collective. However, we notice that for humans, learning from other agents occurs at a much quicker rate when comparing it to learning on your own. For example, it is much easier to learn calculus when it is being taught to you in class/lecture than it is to discover it on your own. Notice that this example can be essentially modeled as a sparse reward function, one with which a traditional reinforcement learning agent would have a tough time given the nature of the exploration problem in RL.

The most common way to transfer this knowledge for human agents is often through learning in school or university. Upon analysis of these academic institutions, we can see that the main tools they provide to aid in this transfer of knowledge is a well designed curriculum and a series of suboptimal demonstrations at each level of said curriculum. This paper attempts to see if these similar learning techniques will aid in the convergence of traditional RL agents. In a series of experiments we compare the results of RL agents with and without the aid of an assistive curriculum on sparse reward environments. We go further and investigate how the rewards and time to convergence improve by preconditioning the models with suboptimal demonstrations at each level.

In the end, the results of our experiments showed that adding a curriculum greatly aided in the agents convergence to the optimal solution. When a set of suboptimal demonstrations were used to precondition the neural networks at each level, it was observed that benefits of the curriculum were magnified. This result opens the door to an incredible new field of study into designing curriculums optimally such that the agent learns to do the task as quickly as possible.

Background and Introduction

Motivation

Often in the field of reinforcement learning (RL), it is reasonable to expect one's rational agents to be just as smart as, if not smarter than, humans. Hence, for certain tasks, we strive to have computational systems simulate the decision making process of adults. In the latter half of the twentieth century, Alan Turing devised a sophisticated approach to simplifying the problem: "Instead of trying to produce a program to simulate the adult brain, why not try to produce one that simulated the child's? If this were subjected to an appropriate course of education, one would obtain the adult brain." This notion was essentially the basis for the field of deep reinforcement learning, where we start off with an overparameterized neural network filled with random initializations. Such a net would essentially represent a mind that is at an initial "blank slate" much like the brain of a child. After the initialization, we subject the agent (based off of this neural net) to experiences in its environment, where it learns to maximize its potential rewards.

Connection to Human Agents

Although the true definitions of what constitutes a "reward" to our brain is unknown (i.e. different experiences incite different levels of reward for different people), this notion is quite similar to how humans learn. However, when humans learn, they often are guided by society. Whether it is a parent holding their hand when they begin learning to walk or when a driving instructor demonstrates how to properly merge onto a highway, the human agent can use this help given by the "agents" around it to optimize rewards, and do so much quicker and more efficiently. This is one of the many reasons why humans were able to be the most successful agents in the current environment: we are able to pass down wisdom and learnings across generations much more quickly than having new agents learn on their own. For example, despite taking the world's most brilliant minds years to discover calculus, the subject is now taught to 16 year old students around the world.

One of the most widely used methods of disseminating knowledge to the future generations is the use of schools and universities. From the perspective of RL, what use does schooling directly provide to a young human agent. In essence, the answer comes down to two main things: a well designed curriculum (consider how we must learn arithmetic before algebra before calculus), and suboptimal demonstrations (teachers often do and present example problems on the board). Given this information, it is rational to believe that these techniques would help guide traditional RL agents to find more rewarding solutions and converge much more quickly as well. Obviously the scope of “Curriculums and Demonstrations help agents converge quicker and perform better” is much too broad for the scope of this paper. Upon further reading of background research, this paper’s team hypothesized that the best place to test the benefits of Curriculums and Demonstrations (and the place where the technique benefitted RL agents most) was in dealing with the RL exploration problem for sparse rewards.

Exploration in RL

Common exploration strategies in classical RL were mostly along the lines of epsilon greedy exploration, where the agent chooses a random action with a probability epsilon, ε, or along the lines of Boltzmann Exploration where the agent computes a distribution over the action space and then samples from that distribution to determine the action to take. The problem with exploration methods becomes clear in sparse reward Markov Decision Processes (MDPs). If the reward function is extremely sparse, the agent taking random actions--even if they are from a prior belief distribution--is not going to encounter the high reward states within any reasonable amount of time. The question now becomes: how do we get an agent to explore in a way that it can find spare rewards, especially ones that are far from the start state? This is known as the RL exploration problem.

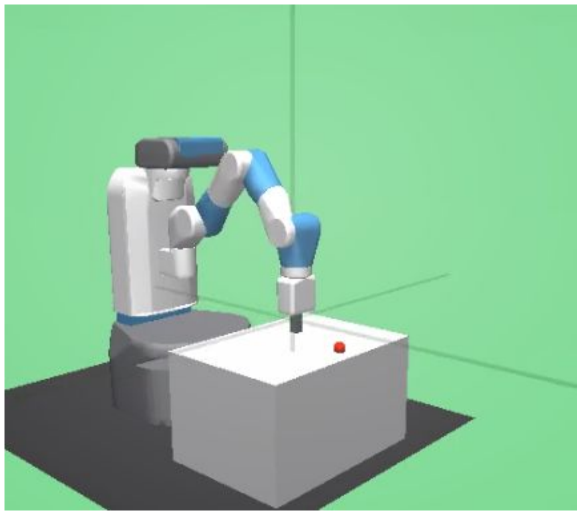
The team hypothesizes that curriculum learning, when combined with demonstration learning, will help RL agents converge quicker. This hypothesis is based on the fact that human agents do this all the time, and we believe this is one of the behaviors that RL agents will share in common with us, humans. It is much easier to solve introductory calculus questions when you have a solid background in mathematics and you have seen a professor do similar problems before in class/lecture. In a more clear example we know that babies learn to walk much earlier when they have the aid of a baby walker.

This paper will test these main hypotheses, along with many smaller scale ones, and attempt to compare vanilla RL methods, with those taught with a curriculum, and those taught with a curriculum + demonstrations. This paper will also touch on the thought process that goes into designing a curriculum, what potential drawbacks this technique has and the reasons behind these.

Methods and Findings

Motivation

The team decided to follow the analogy of the baby that first uses a walker to learn to walk on its own. We decided the best way to translate this analogy into the world of RL is by using a start state curriculum. Essentially, we would modify the environment states such that a high reward signal would be easier to achieve under more traditional exploration methods. As the agent would master getting high returns on the modified environment, the environment would incrementally mold itself back into its original unmodified start state distribution. A point of note is that many other papers have shown Curriculum Learning to work, both in deep learning and vanilla reinforcement learning, but none of these papers seemed to have explored the tremendous benefits a technique like this can provide to MDPs with sparse reward functions nor how suboptimal demonstrations can further help with convergence.

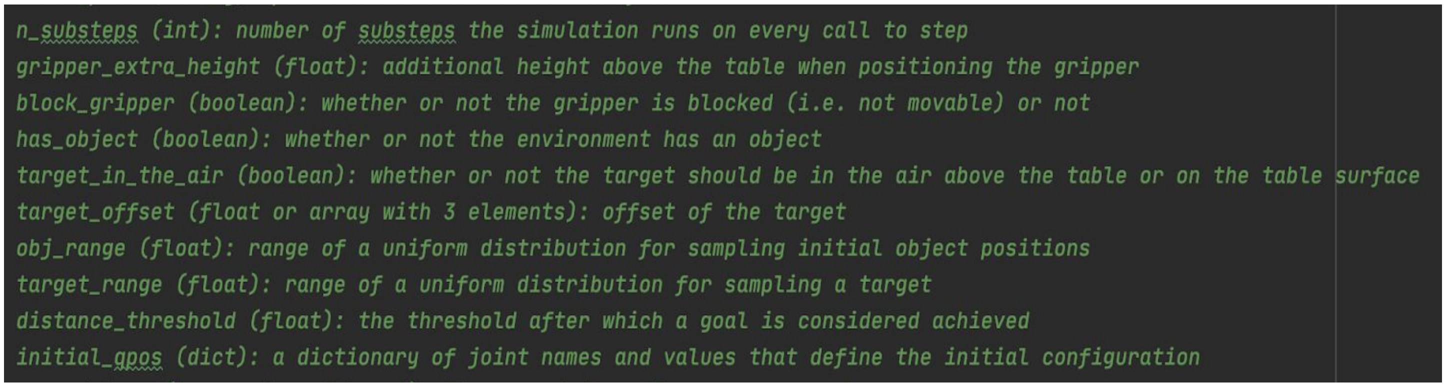


In order to implement this, we initially needed to find a library and a corresponding sparse-reward environment. We needed one that would easily allow us to change the environment to follow a curriculum, all while being easy to view the benefits of said curriculum. The team decided to go with openAI’s gym library, where we could use gym environment wrappers to override the env methods like *observation*, *reward*, and *reset*. Furthermore we decided to use the “Fetch”-sereis environments because they had known sparse rewards, and we hypothesized that a curriculum would be easy to design for one of these. Specifically, our team decided to focus on the ‘FetchReach-v1’ environment.

Environment Information and Changes Made

‘FetchReach-v1’ is an environment in which an agent needs to actuate the tip of a robot-arms gripper to a desired location (the target location for the gripper changes every iteration). For each time step, the environment has a fixed living reward of -1 and a reward of 0 when the gripper is at the goal. Evidently, this is an extremely sparse reward function. We have altered it so that there is a living reward of 0 and a reward of 10 for achieving the agent's end goal. We just did this to amplify the reward signal. Note, this action did not change the sparsity of the rewards. Secondly, we also gave the agent a few “free” timesteps in the beginning where no reward would be counted in order to get to the desired location. Following these free-timesteps, there are a fixed 50 timesteps where the rewards are recorded (0 for not being on the target, and 10 for being on the target, at every timestep). In this modified environment, a perfect agent would use the “free” timesteps to move to the target location, and then spend the remaining 50 timesteps holding the gripper at said location, receiving a total of 500 points.

Lastly, the reset method of the environment was overwritten to allow for constraints to be passed in, ensuring that the start state was consistent with the difficulty prescribed by the curriculum. Taking this a step further, we were able to change the environment such that we read a curriculum from a json file and then run a training loop according to that curriculum. Here are some of the constraints we were able to change in our curriculum:



```
n_substeps (int): number of substeps the simulation runs on every call to step
gripper_extra_height (float): additional height above the table when positioning the gripper
block_gripper (boolean): whether or not the gripper is blocked (i.e. not movable) or not
has_object (boolean): whether or not the environment has an object
target_in_the_air (boolean): whether or not the target should be in the air above the table or on the table surface
target_offset (float or array with 3 elements): offset of the target
obj_range (float): range of a uniform distribution for sampling initial object positions
target_range (float): range of a uniform distribution for sampling a target
distance_threshold (float): the threshold after which a goal is considered achieved
initial_qpos (dict): a dictionary of joint names and values that define the initial configuration
```

Curriculum Development

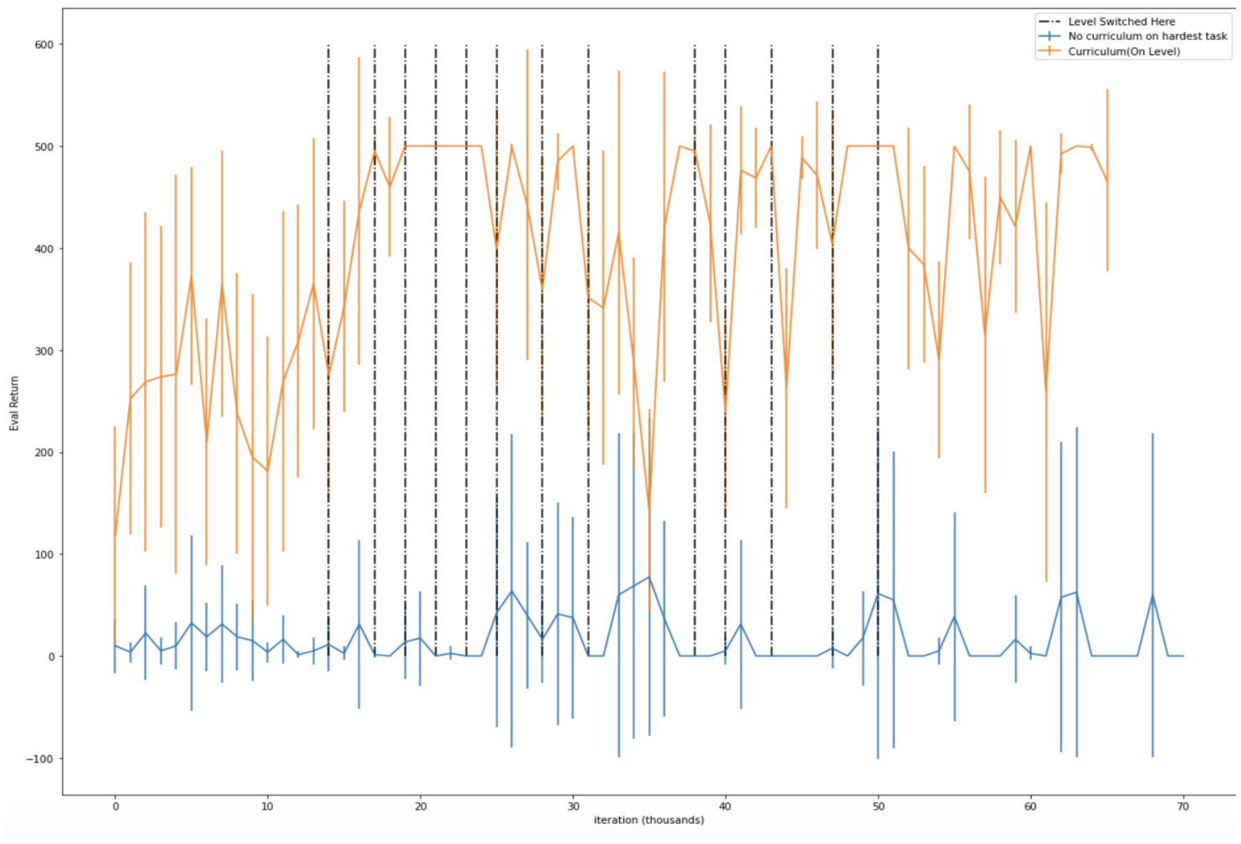
Over the course of our experiments, there were many things that we learned to consider when designing curriculums. This will be discussed in detail in the Discussion and Applications section but we are going to note a few major ones here to motivate why the curriculum(s) used were chosen. When designing the curriculum, it is imperative that the levels are spaced out enough that the agent gains new experiences and learns to fit to the new states in the current level. On the other hand, it is also important that the agent is not spending too much time on a single level, overfitting to it, such that when it transitions to a new level, it can no longer fit to the new states presented. Keeping this in mind, the curriculums designed in the following experiments are designed to automatically jump to the next level, once an agent can consistently get a reward higher than a certain threshold (labeled the 'reward_threshold' in the curriculum json files). It was also ensured that the last level of the curriculums was the same as the unconstrained environment. Below are examples of curriculum that were developed for our environments:

'FetchReach-v1' :

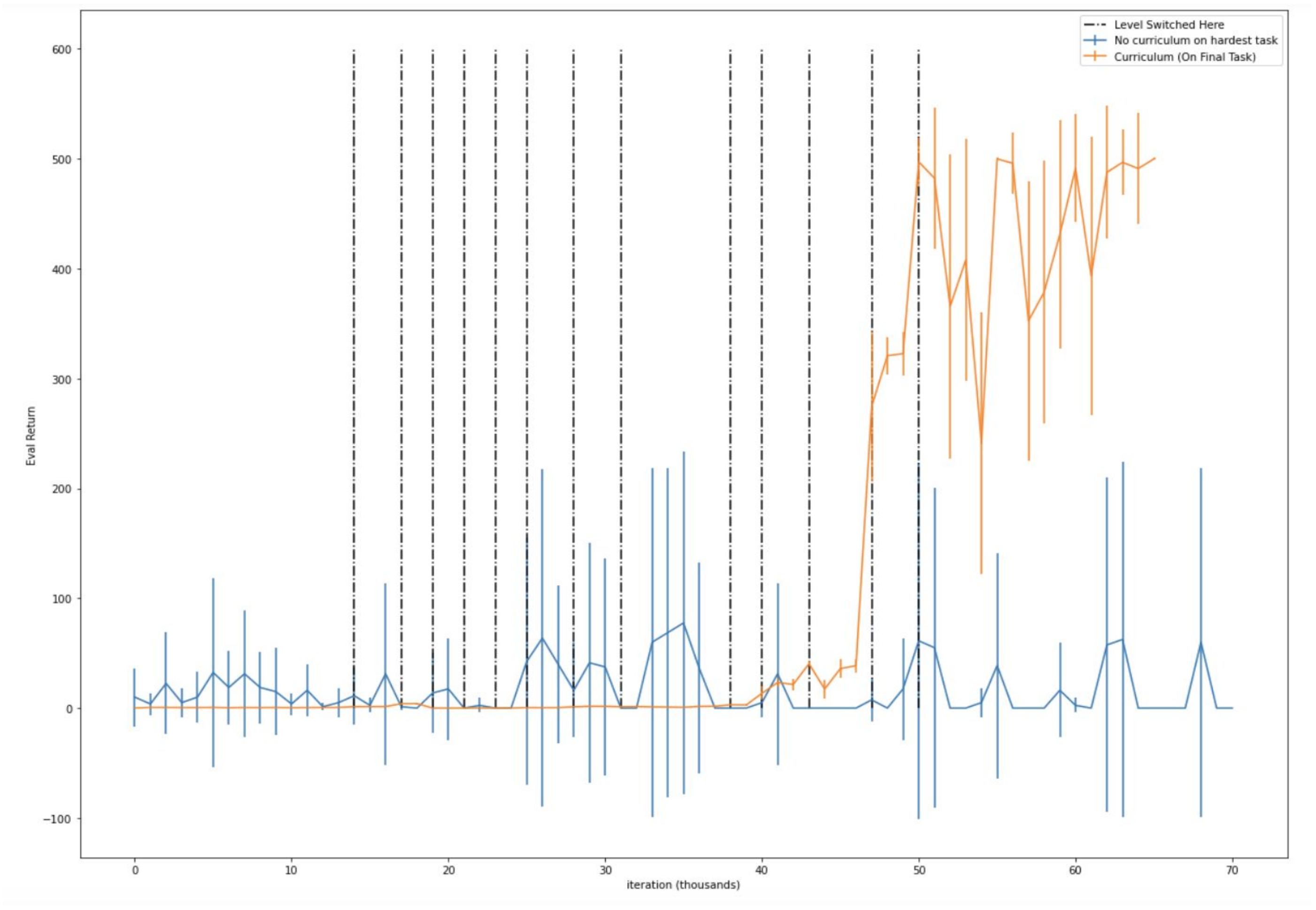
```
{
    'reward_threshold':     [300, 300, 400, 400, 400, 400, 420, 420, 420,
420, 420, 420, 420, 420],
    'target_range':         [0.05, 0.05, 0.07, 0.07, 0.07, 0.09, 0.09,
0.09, 0.09, 0.11, 0.11, 0.13, 0.13, 0.15],
    'distance_threshold':   [0.13, 0.11, 0.11, 0.90, 0.50, 0.50, 0.30,
0.20, 0.2, 0.20, 0.10, 0.10, 0.05, 0.05]
}
```

Note that in all of the curriculums, the max reward possible is the same at every level,500.

Curriculum Learning Results



As can be seen in the figure above, the agent placed on a curriculum performed much better on the final level (aka the unmodified environment) than the agent trying to learn from scratch. Here the orange line represents the return of the curriculum assisted agent *on the current level* of the curriculum. The vertical dashed lines represent the points where the curriculum decides to transition the agent to the next level. It can be observed that after a level change, the agent's performance drops, and then regains its rewards as it learns to fit to the new states present in the current level. In the figure below we see how the agent performs *on the unmodified environment* as it progresses throughout the levels. We can see from this graph that the agent placed on a curriculum successfully completes the final task by the end. However, the lone agent is not even close to converging



As can be seen, the curriculum agent doesn't really begin posting returns until just a few levels before the end. This is because of a parameter in our curriculum called 'distance_threshold'. When this is high (like it is in most of the beginning levels), it means the threshold for the goal to be considered achieved (i.e. the arm is *close enough* to the goal such that a reward is given to the agent). Hence, before the last few levels, the arm is never brought “close enough” to be considered “a goal state” in the eval environment, causing the lack of returns in the early levels seen above

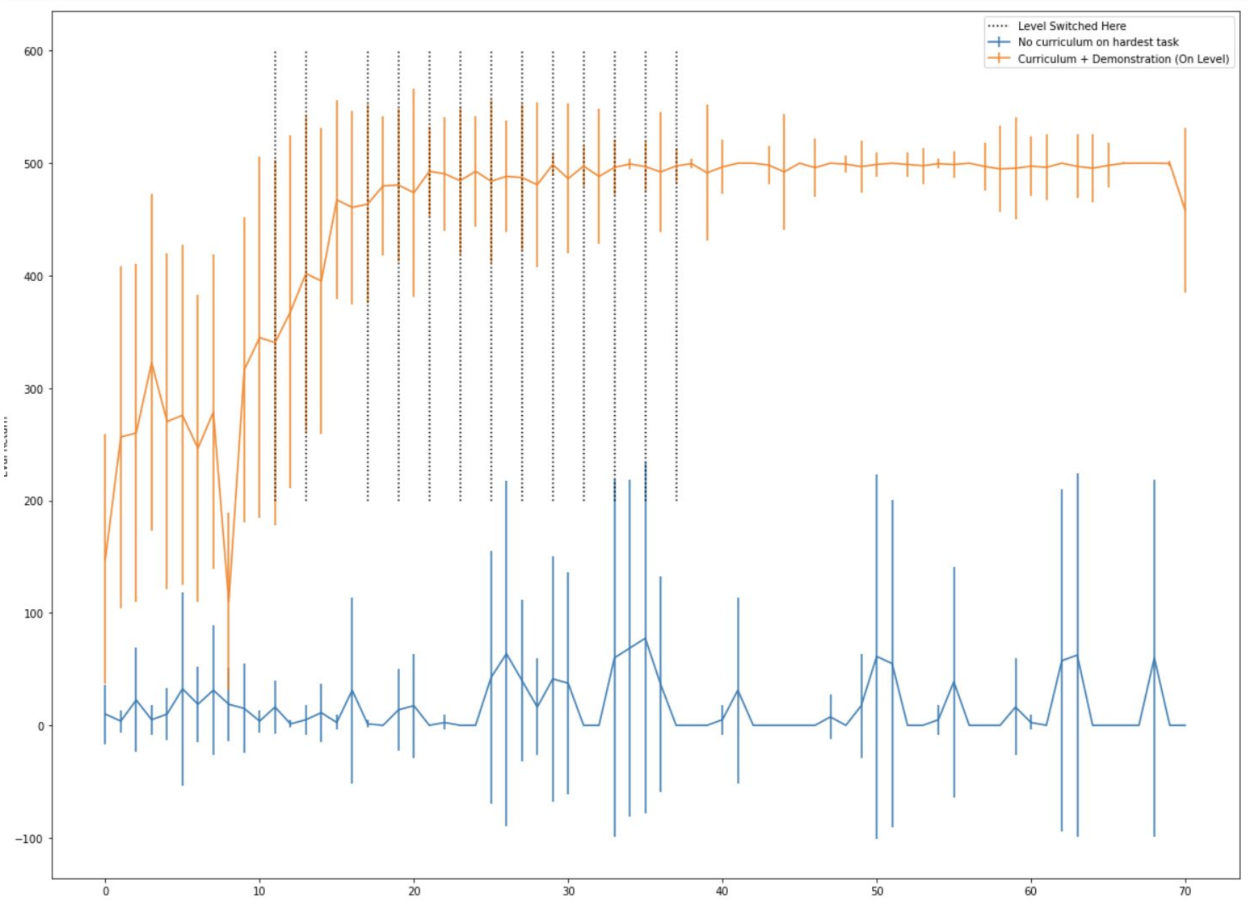
Incorporating demonstrations

While training the agents described above, we were also able to record demonstration data from the evaluation episodes, capturing multiple runs for every level along the curriculum. Note that the data collected is collected at every level, and is indeed suboptimal. The team hypothesized that incorporating this demonstration data along every level of the curriculum would improve the performance even further. To do so, we decided to use a similar method to DfQD, where the net is trained in two phases.

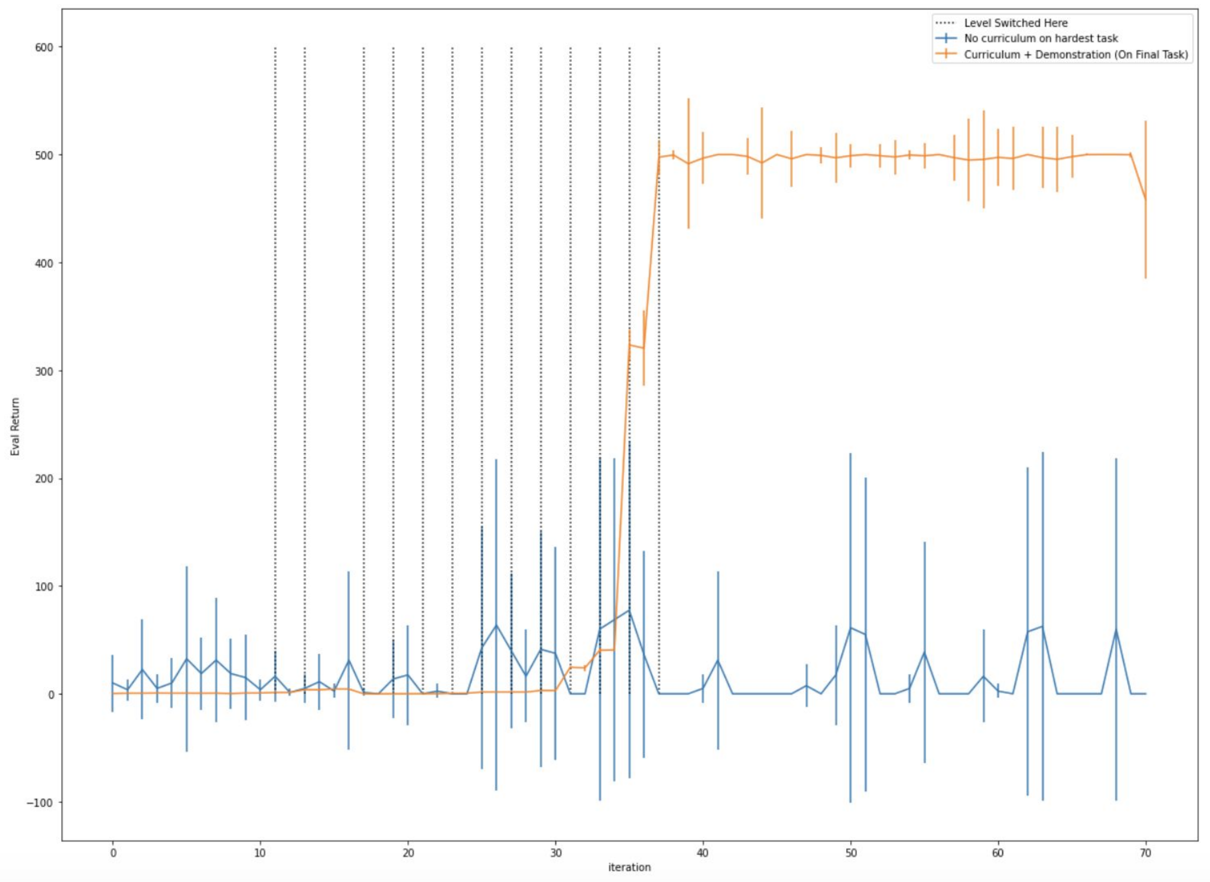
$$J(Q) = J_{AC}(Q) + \lambda J_C(Q)$$

The above loss function is used to train the net. Here, $J_{AC}(Q)$ represents the regular Actor Critic loss being optimized by the agent, whereas $J_C(Q)$ is the classification (like the one from imitation learning), which is included when training on a sample from the demonstration data. In the first phase, the net is trained on the demonstration data using imitation learning to optimize $J(Q)$. In the second phase, the agent is allowed to interact with the environment, adding states to its replay buffer, where the demonstration data is also present. The agent is then trained to optimize $J(Q)$, where the $\lambda J_C(Q)$ term is only included on the demonstration transitions. Over time, the λ term is decayed, eventually leaving us with just the regular actor critic loss. The decaying λ is placed there in order to prevent conflicts in gradient optimization. In essence, although J_{AC} and J_C both have the same optimum, they are not the same loss function, and we want to use J_C as a boost for the agent to eventually optimize J_{AC}. Hence we want to use a decaying λ to allow a smooth gradient transition between J_C and J_{AC}.

Demonstration + Curriculum Results



The graphs above and below follow the same format as the corresponding graphs in the Curriculum Learning Results subsection. Comparing the above and below graphs to the ones from the previous subsection, it can be seen that incorporating demonstration allowed the net to converge in fewer iterations, and the final solution seems to be much more stable (less deviations from a perfect score are seen). Once again, note that at each level the agent is only presented demonstration data from the current level or lower. This was done to stick to the theme of how humans would learn in school or university. When learning algebra, we can only view suboptimal demonstrations of current level or lower; we are not given demonstrations of calculus in algebra class. Below we can once again see the lack of Eval Return on the early-middle levels for the same previously mentioned reasons. However, notice that in the late levels, the growth of the agent is much more pronounced and stable compared to the corresponding growth of a curriculum agent without demonstrations.



Further Discussions & Applications

Discussion of Key Observations

As previously mentioned, it is clear from the above results that our team's hypotheses were correct. It is definitely the case that Curriculum Learning did in fact improve the convergence of the model drastically, and truly was a great way to tackle the RL exploration problem in sparse environments. In fact, in multiple experiments, the control RL agent was left to train for upto five million iterations, and the agent had still not converged to a stable high score. Out of all our experiments, a control agent using traditional exploration was never observed to converge. Without incorporating a curriculum, the team was in fact unable to get the vanilla RL agents to converge to a consistent solution using traditional actor critic exploration methods. Based on the confirming experiments from multiple trials, it is clear that Curriculum Learning worked incredibly well in finding a solution to this very sparse reward function.

To touch upon the evaluation task performance further, and how it seems to jump quickly over just a few levels, the team actually found this occurrence extremely odd at first. It seemed that the agent was not learning at all in the early levels of the curriculum, or perhaps even "anti-learning", where it would perform worse than even the naive agents on that level. However, looking at the per-level performance of the curriculum agents, it is made clear that a lot of learning is actually taking place in these early levels. Furthermore, the team also tried to remove many of the early levels where it seemed as though learning was not taking place. Removing many levels resulted in agents that did not show improvements compared to the control.

Furthermore, we see an even greater improvement in returns when a possibly suboptimal demonstration is added. The team did not expect such a drastic improvement with the help of only a handful of episodes of demonstration data, but the data was clear in this result, and was conclusively confirmed over multiple trials. This confirmed our hypothesis that combining Curriculum Learning and Demonstrations--into a method that the team likes to call *Scholastic Learning*--like humans do in schools and universities, immensely improved performance, and perhaps even lead to taking the agent's average returns to heights it would never have converged to previously.

Many key points of interest regarding our tinkering with curriculums were noted throughout the experiments that have not been shared already through the paper. The first is that tweaking the design of the curriculum is extremely important. As previously mentioned, this is because the curriculum works best when the agent spends 'just the right amount of time' on each level. If the agent spends too little time, then the policy is not permitted the chance to fit correctly to the set of new states presented in the current level. If the agent spends too much time on a level, then the agent's parameters overfit to the current start states, and is not flexible enough to transition to the next level while maintaining returns. Strangely this effect can be both mitigated and exacerbated by complexity of the model. If the model has too many parameters, it is actually more sensitive to overfitting to the current level. However, if we decrease the model's complexity--note that this causes no issues as the inherent model for Fetch-Reach is a simple one: if the target is to your left, move left, etc--the model is far more generalizable to unseen data and start states. Hence, it is less likely to fall into this "overfitting" trap on each level. By incorporating this, the model's sensitivity to 'imperfections' in the curriculum is dramatically reduced. Furthermore, the team found that this curriculum learning method can be stacked on top of any existing RL algorithms. Although only actor-critic and DfQd like methods were tested in this paper, the team did try this with other model-free techniques, including pure imitation learning, and it seems to have shown promising positive preliminary results (but this is for the scope of another paper).

Future Works and Applications

The team believes that these results open the door to a plethora of future interesting works. In this project we were able to show the tremendous potential of including curriculums and demonstrations for sparse rewarded functions. Most obviously, there is a lot more work to be done in the field of designing curriculums. We were able to discuss patterns that the team saw in the relative sensitivity of models to their respective curriculum. however this seems to be just a few of the potentially many trends that surface in Curriculum Learning. The same can be said for incorporating demonstrations along with a curriculum.

However, this can open a few completely new problems in Reinforcement Learning. Consider an RL agent that--given the current performance metrics, curriculum information, and properties of the environment of *another agent*--is trained to decide when the latter agent is ready to transition to the next level in the curriculum. This agent could be trained to optimize the amount of training time along each level, mitigating the sensitivity problem of spending 'just the right amount' of time on a level. A similar thing can be done with an RL agent which, given some imperfect details of the nature of an environment and the observation+action spaces of a *different potential agent*, could even design a curriculum for said agent.

Furthermore, techniques like this could be incredibly beneficial to transferring knowledge from an old agent to a new agent. Consider the case where a robotic arm with four degrees of freedom is working to do a task (a task that it was trained to do using RL). Now however, let's say an engineer hypothesizes that this same task (i.e taking place in the same RL environment) could be done more efficiently or optimally using an arm with 7 degrees of freedom. How do we transition this knowledge between two agents that have completely different action spaces? The answer could lie in extending the techniques of this paper. It could be possible to design an RL algorithm (or frankly any other method) that can take information from the policy of the old agent, and design a curriculum for the new one. This will allow the new agent to have a jump start in learning compared to training it from scratch. Once it completes the curriculum, it can continue to learn and discover further optimizations from its extra degrees of freedom. Just like humans, who transfer knowledge from generation to generation, a technique like this could be used to transfer knowledge from an old agent to a newer, upgraded agent. A similar trick could potentially be done if the agent (for example, the same 4 degree of freedom robot) is transferred to a closely related but different environment (i.e. a different part of the assembly line).

As the above examples illustrate, this set of experiments and the potential future research (if successful) could have a ton of real world applications when deployed in everyday life. For example, in order for *customized* robot agents to be deployed into a consumer's personal life (for example, a robot arm that learns to cook in *your existing* kitchen to optimize *your* tastes) there needs to be a way for an agent to learn rather quickly. Transfer learning is a potential solution to this problem, but still requires many episodes to complete the transfer of knowledge. If this could be stacked upon an optimized curriculum for the agent (through one of the potential techniques mentioned above) it may bring us many steps closer to making this notion a viable future

Conclusions

All in all, we observe that applying Curriculum Learning to Reinforcement Learning problems with sparse reward functions can have a great impact on their efficacy and convergence time. Results from this paper indicate that, similar to human agents, if the agent is given demonstrations at each level of the curriculum, this greatly increases positive impacts. Also notice that the better designed the curriculum is, the more effective it is at speeding up the agents convergence--similar to how better schools may provide a higher quality of education. These results open up a whole new field of best practices to creating effective curriculums, something that was briefly discussed in this paper. Furthermore it opens the door to potentially tremendous applications in increasing the feasibility of RL agents in the everyday world

References

(n.d.). Retrieved from https://danieltakeshi.github.io/2019/04/30/il-and-rl/

-, P. E., By, -, & Staff, P. E. (2018, July 18). Extending OpenAI Gym environments with Wrappers and Monitors [Tutorial]. Retrieved from https://hub.packtpub.com/openai-gym-environments-wrappers-and-monitors-tutorial/

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*. doi:10.1145/1553374.1553380

Czarnecki, W. M., Jayakumar, S. M., Jaderberg, M., Hasenclever, L., Teh, Y. W., Osindero, S., . . . Pascanu, R. (2018, June 05). Mix&Match - Agent Curricula for Reinforcement Learning. Retrieved from https://arxiv.org/abs/1806.01780

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., . . . Gruslys, A. (2017, November 22). Deep Q-learning from Demonstrations. Retrieved from https://arxiv.org/abs/1704.03732

Matiisen, T., Oliver, A., Cohen, T., & Schulman, J. (2017, November 29). Teacher-Student Curriculum Learning. Retrieved from https://arxiv.org/abs/1707.00183

Moreno, A. I. (2019, December 15). Reinforcement learning framework and toolkits (Gym and Unity). Retrieved from https://towardsdatascience.com/reinforcement-learning-framework-and-toolkits-gym-and-unity-1e047889c59a

Seita, D. (n.d.). AWAC: Accelerating Online Reinforcement Learning with Offline Datasets. Retrieved from https://bair.berkeley.edu/blog/2020/09/10/awac/

Weinshall, D., Cohen, G., & Amir, D. (2018, June 08). Curriculum Learning by Transfer Learning: Theory and Experiments with Deep Networks. Retrieved from https://arxiv.org/abs/1802.03796

Weng, L. (2020, January 29). Curriculum for Reinforcement Learning. Retrieved from https://lilianweng.github.io/lil-log/2020/01/29/curriculum-for-reinforcement-learning.html