RL Research Diary

Hager Radi

20-26 January 2020

19 January 2020

I used to think that continuing tasks can be modeled as episodic tasks only if we added a few assumptions on how to define a terminal state. It is easier to handle episodic tasks in reinforcement learning(RL), so what might stop us from modeling all continuing tasks as episodic tasks, if we apply the following criteria? Continuing tasks can be cut into episodes by defining an interval over the time steps we have. This makes all episodes have the same length, but this is not an issue. A terminal state is any state at the end of the time-interval we specify when designing the task. Each new episode would start from the last state in the previous episode. However, this is violating the assumption of episodic tasks where a new episode should start from a previous episode. For the reward, we can maximize the total discounted reward in a time-frame we defined as an episode. This is not fully true as it does not necessarily mean maximizing the overall average reward. We can use a discount factor of 1 instead, and maximize the average reward in each episode instead. Does this make it closer to our main objective in an episodic task? I believe some tasks can survive under these assumptions while they make it feasible to use algorithms originally defined for episodic tasks. -

In the real world, humans observe data in sequences. In supervised learning, we shuffle the data when optimizing our models. In reinforcement learning, observations are mostly shuffled in batches during policy updates. We treat our data as individual frames losing the notion of time. If we are trying to learn how to build computational agents that are similar to human minds, why do we change the way we learn from the world data and follow a direction that is against human nature? Optimization methods such as gradient descent rely on learning from shuffled data; they do not learn efficiently if data is not shuffled. This might be a special case in supervised learning when the model receives the same samples again and again, but even in reinforcement learning, we never treat our data as sequences, which might be of a huge impact. In RL, we are trying to learn a mapping from situations to the actions taken by the agent. It should be easier to learn from an interval of data, rather than single frames. Multi-step prediction and control methods such as n-step TD and SARSA rely on more frames of data for function updates. However, policy-based methods who rely on multi-epochs tend to shuffle the data, and destroy its natural ordering. I have been thinking of a way to treat data as sequences, in their original order as received from the environment. We can introduce a recurrent neural network where data representations can be learned in a sequential way. This might be an addition to the idea of learning representations of situations rather than single states. By this, we can contribute to better planning since learning is more dependent on long term time frames. Does this contribute to learning a better model of the environment? I believe so. Sequence-based states will formulate a more accurate version of the world we are trying to model.

21 January 2020

Transfer learning proved to be of a high impact in supervised learning, and other AI algorithms. For RL agents learning similar tasks, can we apply transfer learning and initialize such agents with the value functions or policies learned by other agents? Knowledge transfer can be helpful for RL tasks as it adds a huge factor of exploration by initializing some states with values already learned for another similar task. It might also be advantageous if one of the sub-problems of the current task lies in the space of the problem solved by the main task. However, this introduces a few challenges. First, it can magnify the problem of catastrophic interference, as it implicitly introduces a much larger state space, and agents can easily forget what they have learned if new data is not similar. Secondly, transfer learning might deviate the agent from learning the optimal solution and lead it into to reach some local minimum because of the optimistic initialization used in some states rather than others. One main principle in learning general-purpose RL agent is learning from scratch rather than relying on any other resources such as humans when learning from demonstrations or other trained RL agents. In this case, knowledge transfer would be one direction against learning general-purpose AI agents. Moreover, it adds an extra challenge to the reproduciblity of RL solutions since they become more dependant on external knowledge. If we do not put a huge weight on reproducibility and the learning from scratch hypothesis, knowledge transfer can still be a step on the way to general-purpose solutions by learning complex tasks through multiple simpler tasks. Can transfer learning have more impact on value-based methods rather than policy-based methods? Is one of them more sample-efficient than another?

23 January 2020

In our class discussion, we discussed the dimension of training feedback in AI solutions where we have instructive vs. evaluative algorithms. Supervised learning (SL) is instructive and reinforcement learning (RL) is evaluative. Both SL and RL lie under the associative category, where the learning maps certain situations. Multiple solutions such as function optimization and bandits lie under the non-associative evaluative feedback. To better understand this categorization, I have been thinking, does there exist a learning algorithm that is instructive and non-associative at the same time? If we want to fit unsupervised learning in our categorization, where will it be? I believe unsupervised learning, where we try to learn some internal structure of our data, follows the evaluative feedback approach rather than correct labels/instructions. It learns how to improve this structure based on the feedback it receives from the environment. It is non-associative in the sense that it finds patterns among the data it represents so the learning does not involve acting in different situations. I cannot come up with a solution/learning technique that can be instructive and non-associative. What about learning by demonstrations and imitation learning? These methods involve a human having the right decisions and correct behavior for the agent to follow. This takes it outside the evaluative feedback category, and makes it more similar to the instructive supervised learning. It is also associative such that it links certain situations with certain actions to make. I believe If instruction involves giving true labels/feedback, it implicitly associates learning with certain situations, and makes it associative by default.

24 January 2020

In Reinforcement Learning, catastrophic interference/forgetting is a big issue that has not yet been researched extensively. It is the tendency of a neural network to forget what it has learned upon learning new information. My hypothesis is that the interference can be solved if we learn how to decompose the original task into sub-goals that contribute to the end-goal. Can an agent learn to decompose the main goal as it observes more states? Consider a robot learning how to score a goal in a football match; it does not learn this directly but rather learn how to get the ball, how to face the goal area, and how to move forward. Learning sub-goals can help learn better state representations, and it can better help with planning. If we focus on the representation part, can the agent follow some sort of unsupervised learning where it distinguishes between these goals so it can avoid catastrophic forgetting in the future? What if the agent learns to cluster the states it gets into different patterns of features unique to each sub-goal? This means an agent is encouraged to optimize for these feature representations, and incrementally can learn the underlying patterns/structure specific to each sub-goal. This cannot be considered a part of the value function approximation. We need to introduce some unsupervised learning approach where representations can be learned incrementally with each example we get. We can introduce an auto-encoder, which runs in parallel with our regular function approximation. The role of this auto-encoder is come up with the unique representation we have mentioned above. When we optimize either for our policy based or our value based algorithms, we use the learned representations rather than using the raw states in our approximations. This is still missing the framing of the loss function to be used, and how frequent this auto-encoder shall be updated. It also introduces the problem of having lossy/inaccurate representations learned by the auto-encoder if we have large state space. It is worth further investigation whether this is still helpful to catastrophic forgetting or not, even with the lossy representations.