Paper: *World Models*

Nirajan Koirala

CSC 9010_001

Summary

Authors explore building generative neural network models of popular reinforcement learning environments in this paper. They first provide a brief discussion of how the understanding of predictive model inside human's brain take place by brains predicting the future sensory data given our current motor actions. They mention that in many RL problems, an agent also benefits from representation past/present states and a good predictive model of future. The RL algorithm is however bottlenecked by credit assignment problem so in practice smaller networks are used as they iterate faster to a good policy during training. They propose training large NN to tackle RL tasks by dividing the agent into a large model and a small controller model. They first train a large NN to learn a model of the agent's world in an unsupervised manner and then train the smaller controller model to learn to perform a task using this world model.

The agent is provided with a visual sensory component that compresses what it sees into a small representative code. It also has a memory component that makes predictions about future codes based on historical information and a decision making component which decides what actions to take based only on the representations created by vision and memory components. They provide description of each of their V, M and C models. Basically, V model compresses what the agent sees at each time frame, M model predicts the future and C model is responsible for determining the course of actions to take in order to maximize the expected cumulative reward of agent during a rollout of the environment. A minimal design is chosen for C. But, V and M models are designed to be trained efficiently using backpropagation algorithm so most of the model parameters are kept in V and M. Using this convention they are able to train C more unconventionally since tackling credit assignment problem is very difficult. For optimizing the parameters of C, they've chosen Covariance Matrix Adaptation Evolution Strategy.

In section 3, they describe how they managed to train the agent model to solve a car racing task. Tracks are randomly generated and the agent is awarded for visiting as many tiles as possible in the least amount of time. They discuss about the individual training methods of each of the models. They mention that the world model (V and M) has no knowledge about the actual reward signals from the environment. Its task is simply to compress and predict the sequence of images frames observed and only controller C model has access to the reward information from the environment. In section 3.2, the procedure of their Car Racing experiment is provided. They discuss about the experiment results later and point that handicapping C to only have access to V would not help solve the environment. After combining V and M, controller C gets a good representation of both current observation and what to expect in the future. Giving the agent both access to z_t and h_t greatly improves its driving capability and the agent does not need to plan ahead and roll out hypothetical scenarios of future. Agent is then able to achieve really high scores taking in stream of raw RGB pixel images and it is able to directly learn a

spatial-temporal representation. Also, as their world model is able to model the future, they can have it come up with hypothetical car racing scenarios on its own.

In section 4, authors discuss about the VizDoom Experiment in which they train an agent inside the hallucination generated by its world model trained to mimic a VizDoom environment. The cumulative reward is defined as the number of time steps agent manages to stay alive during a rollout since there are no explicit rewards in this environment. They discuss the procedure of training their model later and mention that M model has to predict whether agent dies in the next frame as well as the next frame itself. After having all the ingredients required to make a full RL environment, they build an OpenAI gym environment and train the agent inside of this virtual environment instead of using the actual environment. While training inside the dream (virtual env), the RNN-based world model is trained to mimic a complete game environment and learns how to simulate the essential aspects of the game like game logic, enemy behavior, physics and 3D graphics rendering. Unlike the actual game environment however, it is possible to add extra uncertainty into the virtual environment thus making the game more challenging in the dream environment. They can control this difficulty using a temperature parameter τ. They also find that the agents that perform well in higher temperature setting generally perform better in the normal setting. Authors notice that their agent is able to discover adversarial policy in the virtual environment governed by M and take advantage of it. They acknowledge their world model will be exploitable by the controller, even if in the actual environment such exploits do not exist. Also, since the controller has access to all the hidden states of M, agent can efficiently explore ways to directly manipulate the hidden states of game engine to maximize the cumulative reward. They discuss about this weakness using previous works and emphasis that to make it more difficult for C to exploit deficiencies of M model, they chose to use the MDN-RNN as the dynamics model which models the distribution of possible outcomes in the actual environment rather than merely predicting a deterministic future. They discuss more about adjusting the parameter τ and provide several instances of using it. It is found that increasing the temperature of the model M makes it more difficult for the model C to find adversarial policies and increasing it too much makes the virtual environment too difficult for agent to learn anything. The scores they receive using this approach tops the OpenAI Gym leaderboard but the variance level is really high.

In section 5, they discuss the iterative training procedure and point that it requires the M model to not only predict the next observation x and done, but also predict the action and reward for the next time step. Later in the last sections, related works in the literature is discussed.

Strengths

- The explanations of different models provided is very intuitive to understand.
- The paper has also provided an interactive version which helps to understand the methods used by having interaction.
- The approach used for training the model using hypothetical scenarios is very clever.

Weaknesses

- The organization of the different sections in the paper could be improved.

Points of Confusion

- I am a little confused about the flow diagram that shows V, M and C together.
- Section 4.5 was confusing and in particular why choosing to use MDN-RNN as the dynamics model makes it more difficult for the C model to exploit?
- What is Covariance Matrix Adaptation Evolution Strategy?

Discussion Questions

- Is the dream training approach that they've used also applicable to train other different kind of games which are inherently difficult to train?
- Is using low τ values a signal that the model they are using is overfitting and getting perfect scores?
- How would a iterative training procedure help models for difficult tasks?