Paper: World Model

Summary

In this paper, the authors train a large neural network to tackle RL tasks, by dividing the agent into a large world model and a small controller model. They first trained a large neural network to learn a model of the agent's world in an unsupervised manner, and then train the smaller controller model to learn to perform the task using this world model. The small controller model lets the training algorithm focus on the credit assignment problem on a small search space, while not sacrificing capacity and expressiveness via the larger world model.

They present a simple model inspired by our cognitive system. In this model, the agent has a visual sensory component that compresses what it sees into a small representative code called as Visual Model (V). It also has a memory component that makes predictions about future codes based on historical information called Memory RNN (M). Finally, the agent also has a decision-making component that decided what actions to take based on the representations created by its vision and memory component called Controller (C). For the Visual Model, they use Variational Autoencoder (VAE); for the Memory RNN, they use Mixture Density Network combined with the RNN (MDN-RNN); for the controller, they use a simple single-layer linear model that maps the input from V and M into action at each time step. This minimal design of C offers some practical benefits. Their V and M models are designed to be trained efficiently with the backpropagation algorithms using modern GPU accelerators, so they would like most of the model's complexity, and model parameters to reside in V and M. The number of parameters in C is minimal in comparison to V and M. This allows them to explore more unconventional ways to train C. They used evolution strategies (ES) to tackle more challenging RL tasks where the credit assignment problem is difficult. To optimize the parameters of C, they chose Covariance-Matrix Adaption Evolution Strategy (CMA-ES).

The first experiment they conducted was a car racing experiment. The agent controls 3 continuous actions: steering, acceleration, and brake. They collected 10,000 rollouts from a random policy and trained VAE (V) model to encode frames into z. They then trained MDN-RNN (M) using z from the V model and also the action from the Controller. Finally, they defined controller outputting action, which changes the environment and is also fed back to the M model. They experiment by using only the V model and the Full World Model (V and M). They found out that with only V model, the agent is still able to navigate the racetrack, however, its motion is wobblily and it misses the tracks around the sharp corners. With both the V and M models, they found out that the motion is much stable, and the agent is able to seemingly attack the sharp corners effectively. Furthermore, the agent does not need to plan ahead and roll out hypothetical scenarios of the future. Since it has access to the information about the probability distribution of the future, the agent can just query the RNN instinctively to guide its action decisions. Their agent is able to achieve a score of 906 +- 21 over 100 random trails, effectively solving the task and obtaining new state of art result. Since their world model is able to model the future, they were able to have it come up with hypothetical car racing scenarios on its own. They can put their C back into the hallucinated environment generated by M. It is almost as if the agent is learning from its dream. They also conducted a VizDoom Experiment, which is basically testing if one can train an agent to learn inside of its own dream and transfer that policy back to the actual environment. In this experiment, they train an agent inside the hallucination generated by its own world model trained to mimic a VizDoom environment. The setup of the experiment is very similar to the Car Racing experiment with a few key differences. Unlike the Car Racing model, for this experiment, their M model will also predict whether the agent dies in the next frame, in addition to the next frame z. This observation, they refer to as done state or d state. After some training, their agent learns to navigate around the dream environment and escape the fireballs. Unlike the actual game environment, however, they can add extra uncertainty into the virtual environment by increasing the temperature T parameter. They found out that the agents that perform well in higher temperature settings generally perform better in normal settings. They finally took the agent trained inside a virtual environment and tested its performance on the original VizDoom environment. The score over 100 random consecutive trails is ~ 1100-time steps, which is far beyond the required score of 750-time steps.

They also realized that their controller model will be able to exploit the world model. The reason for this is because the controller has access to all the hidden states of M and therefore, access to the internal states and memory of game engines, rather than only the game observations that the player gets to see. Therefore, their agent can easily find an adversarial policy that can fool the dynamics model. The saw that increasing the temperature of the M model makes it more difficult for the C model to find the adversarial policies. In their experiment, they used an iterative training procedure to train their model so that their agent will explore its world and constantly collect new observations to improve and refine its world model.

Strengths

- The paper includes very relatable examples to explain concepts. For example, the baseball player analogy.
- The figures included in the paper are also descriptive and easy to understand.
- They also include a summary of the experiments that they did in bullet points. This helps readers skimming through the paper to understand exactly what they did in a single look then having the search around in the paragraph.

Weaknesses

- I felt like they made this paper a bit informal than a standard academic paper. I understand that including the analogies and summary of the procedures makes it easy for the reader to follow the paper.

Confusions

- I want to learn more about what they meant by the credit assignment problem.
- What is a model based RL vs. a model free RL?
- How does evolution strategy work?

Discussions

- How do you think the world model and the controller model interact with each other?
- What does their controller have significantly fewer parameters than V and M? What is the benefit of doing that?
- How good this model replicates our cognitive system? What are some sectors that can still be improved? Is there anything special about a human cognitive system that can never be replicated by such World Models?