World Models Writeup

For SforAiDl Summer Assignment

Arijit Gupta

16 July 2019

Aim of the article

The goal of this article[1] *is to distill several key concepts from a series of papers 1990– 2015 on combinations of RNN-based world models and controllers. In this article, a simplified framework that can be used to experimentally demonstrate some of the key concepts from these papers, and also suggest further insights to effectively apply these ideas to various RL environments is presented*

1 Components of the Agent Model

1.1 Vision Model (V)

The role of the V model is to learn an abstract, compressed representation of each observed input frame that is part of a sequence. A simple Variational Autoencoder(VAE) is used to compress each image frame into a small latent vector z.

1.2 Memory Model (M)

The M model serves as a predictive model of the future z vectors that V is expected to produce using a Mixture Density Network combined with a RNN(MDN-RNN).

1.3 Controller Model (C)

The C model is responsible for determining the course of actions to take in order to maximize the expected cumulative reward of the agent during a roll-out of the environment. C is a simple single layer linear model that maps z_t and h_t from the concatenated input vector $[z_t h_t]$ directly to action at at each time step.

$$a_t = W_c[z_t h_t] + b_c$$

These V, C and M models interact with the environment together to form the agent model[2]. The raw observation is first processed by V at each time step t to produce z_t. The input into C is this latent vector z_t concatenated with M's hidden state h_t at each time step. C will then output an action vector at for motor control, and will affect the environment. M will then take the current z_t and action at as an input to update its own hidden state to produce h_{t+1} to be used at time $t + 1$.

[1] Jürgen Schmidhuber David Ha. *World Models*. `https://arxiv.org/abs/1803.10122v4`. Accessed on 16-07-2019. 2018.

[2] Normally state of the art models have parameters in the order of 10^8 to 10^9 while the training models looked at here, have 10^7 parameters. This is because it is based on model-free RL models which have 10^3 to 10^6 parameters.

2 Testing the model on different environments

2.1 Car Racing Experiment

The agent is tested on a car racing experiment for feature extraction. When the C model is handicapped with access to only V and not M, it resulted in wobbly and shaky driving. On being given access to M as well the driving is more stable. The feature extracted were then used by the M to generate dreams, which were actually its predictions of what the V would produce.

2.2 Doom Experiment

It is also tested on a Doom experiment for learning inside a dream. The procedure is largely the same as the car racing experiment. The key change is that in the car experiment the M model is only predicting the z_t while here the predictions are used to simulate a full RL environment. When the model trains in the simulated environment the V model is not needed to encode pixels, so the model trains completely in a latent space environment. This is done using a iterative training process as follows:

1. Initialize M, C with random model parameters.

2. Roll-out to actual environment NN times. Agent may learn during roll-outs. Save all actions a_t and observations x_t during roll-outs to storage device.

3. Train M to model $P(x_{t+1}, r_{t+1}, a_{t+1}, d_{t+1} \mid x_t, a_t, h_t)$

4. and train C to optimize expected rewards inside of M.

5. Go back to (2) if task has not been completed.

3 Results of the Tests

The main focus was to implement iterative training using the C-M model to learn in dreams and then transferring it to the real world model and it had a good success rate in the above experiments, making it a really exciting topic

References

David Ha, Jürgen Schmidhuber. *World Models*. `https://arxiv.org/abs/1803.10122v4`. Accessed on 16-07-2019. 2018.