Duckietown - Reinforcement Learning based on World Models
Duckietown - megerősítéses tanulás a World Models alapján

conDUCKtors
Máté Büki, Péter Zoltán Sánta, Gábor Tamás
Deep Learning in Practice with Python and LUA
VITMAV45
Budapest University of Technology and Economics
2019/20/1

Abstract

In our homework, we use a reinforcement learning framework to teach an agent to be able to follow the lane in the Duckietown Gym environment. The model our work is based on was named World Models (1) and was published in 2018. The main idea of the model is to use a compressed representation of the observable environment instead of the raw image seen by the agent. The transformed observations are then passed to a recurrent neural network, which is responsible for the prediction of the next observation. Finally, using reinforcement learning, we train our agent according to the so called SAC policy.
Házi feladatunk keretein belül egy megerősítéses tanulás alapú, három komponensből álló modellt használva tanítottunk egy ágenst arra, hogy képes legyen az út követésére a Duckietown Gym szimulációs környezetben. Ehhez munkánk során a 2018-ban publikált ún. World Models(1) modellt használtuk. A modell különlegessége, hogy nem az ágens által látott környezet nyers képét használja fel, hanem a környezet egy tömörített reprezentációját. Az így kapott, már transzformált megfigyeléseket továbbítja egy rekurrens neurális hálónak, amelynek feladata, hogy az ágens által a következő lépés során látott környezet transzformált formáját prediktálja. Végül az ágensünket megerősítéses tanulás segítségével, a rekurrens neurális háló kimenetét felhasználva tanítjuk az ún. SAC policy alapján.

1 Introduction

Many of the algorithms in the field of artifical intelligence are inspired by how humans learn and think. Our brain is able to learn an abstract representation of the information it gets from our environment (2), (3). Our homework is based on a model called World Models(1), which mimics this behaviour. In their article, the authors present a new framework for facilitating the policy evolution in a reinforcement learning task. The main idea of their work is that they transform the environment seen by the agent into a latent vector, and the reinforcement learning algorithm is trained based on the already transformed reality. So the agent learns in its own world, hence the name of the model.

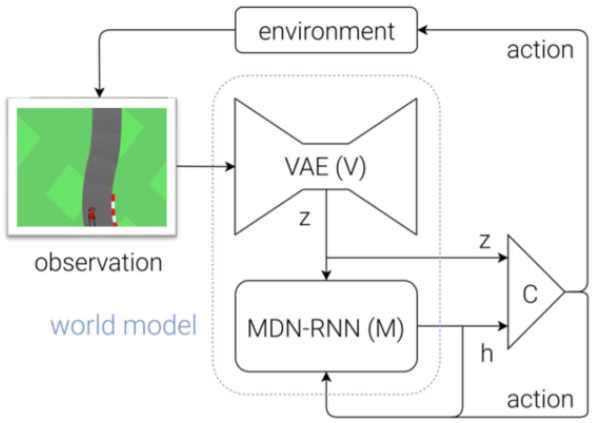

Figure 1: Components of World Models

2 Agent Model

Our task was to implement the World Models(1) (that has been already done for the OpenAI CarRacing environment) in the Duckietown Gym environment. In this chapter we briefly introduce the architecture of the model.

World Models has 3 basic components, depicted in figure 1.

2.1 Vision (V)

The Vision model is a simple Convolutional Variational Autoencoder (VAE) (4), (5) which takes an observation (2D image) from the environment and encodes it into a latent vector z of size 64.

The architecture of the VAE network is shown in figure 2.

2.2 MDN-RNN (M)

The second building block of the model we used is an LSTM(6) recurrent Neural Network with Mixture Density Network(7). It takes the latent vector z from the vision module, the previous action a chosen by the controller and the previous hidden state h of itself. Similarly to the vision module, its goal is to capture a latent understanding/observation of the environment by predicting what would the next z look like.

The architecture of the Memory RNN network is shown in figure 3.

2.3 Controller (C)

In the original implementation it was a simple single layer linear model which maps the vision module's observation z and the memory module's hidden state h to an action a at each time step. The output is an action-vector (numpy array) of size 2 containing the quantitative representation of the velocity [-1.0 to 1.0] and the angle of the steering wheel [-1.0 to 1.0]. Instead of the simple linear model, we used the so called SAC reinforcement learning policy (8). SAC uses entropy regularization and is trained to maximize a trade-off between expected return and entropy. Entropy is used to describe the randomness in the policy. A higher value of entropy leads in more exploration, which can accelerate the learning process.

3 Implementation

In our homework, we used the Duckietown Gym (9) environment. Gym (10) is a toolkit for developing and comparing reinforcement learning algorithms. It supports teaching agents everything from walking to playing games like CarRacing or Pacman. Duckietown is a playful learning experience

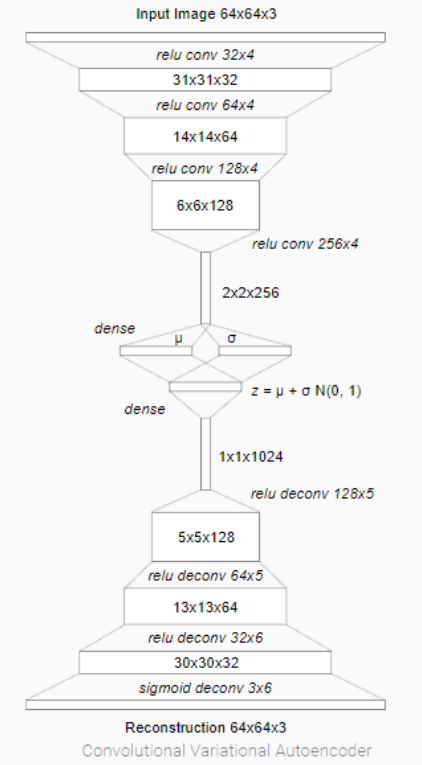

Figure 2: Architecture of the VAE network

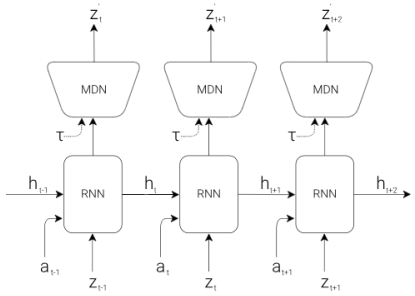

Figure 3: Architecture of the Memory RNN network

for state-of-the-art robotics and AI. It makes it possible to develop and test autonomous driving algorythms with Deep Learning in a joyful sandbox environment. To make the development easier, the Duckietown Foundation published the Duckietown Simulator based on the OpenAI Gym environment.

Modifying the structure of the model presented in (1) was out of the scope of our homework. Our goal was to modify an already existing PyTorch implementation of World Models[1] so that we can apply it in Gym-Duckietown.

3.1 Data preparation

We have created a dataset of 100 training, 50 testing and 20 validation rollouts. A rollout consists of four arrays,

[1] https://github.com/ctallec/world-models

3.2 Training the model

Since our model has three components, we had to train it sequentially. In the first step, the VAE network was be trained independently in the following way:

1. The network takes an image with a size of (640 x 480 x 3) as an input.
2. A convolutional network creates a so called z vector of length 64 as a compressed representation of the input image.
3. A deconvolutional network then takes the z vector as input and generates a (640 x 480 x 3) sized image as an output.
4. Finally, the VAE network computes the loss based on the similarity of the input (original) and the output (generated) images. If the two images are quite similar, that means that the VAE learned to compress images efficiently into a vector of length 64.

Note that the deconvolutional network will only be used during training, after that it will be "deactivated".

Before making a step forward, we will first fix the parameters of the VAE (we do not want it to train any more). Then the RNN takes two inputs in every step: the current z vector from the VAE and the previous action a chosen by the controller.

During the training of the RNN, we will use a simple random number generator (that generates two numbers in the range of [-1.0, +1.0]) to simulate the controller (due to the fact that it will not have been trained yet). The output of the RNN is the predicted z value of the next step ($z + 1$), so that the loss will be calculated based on the similarity of the current output of the RNN and the input from the VAE in the next step.

Finally, we will train the controller, but before that, we will first fix all the parameters of both the VAE and the Memory networks.

The controller takes two vectors as input, one from the VAE (the current z vector) and one from the Memory (the hidden h vector) and generates an action a as an output.

Based on the action of the controller, the environment generates a reward (a float number) that indicates how good decision the controller has made. The aim of the controller is to maximize the reward it gets.

3.3 Evaluation

4 Summary

Acknowledgments

We wish to acknowledge the help provided by Róbert Moni, who has guided us throughout the semester.

References

References

[1] D. Ha and J. Schmidhuber, "World models," *arXiv preprint arXiv:1803.10122*, 2018.

[2] N. Nortmann, S. Rekauzke, S. Onat, P. König, and D. Jancke, "Primary Visual Cortex Represents the Difference Between Past and Present," *Cerebral Cortex*, vol. 25, pp. 1427–1440, 12 2013.

[3] G. W. Maus, J. Fischer, and D. Whitney, "Motion-dependent representation of space in area mt+," *Neuron*, vol. 78, no. 3, pp. 554–562, 2013.

[4] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[5] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," *arXiv preprint arXiv:1401.4082*, 2014.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] C. M. Bishop, "Mixture density networks." 1994.

[8] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.

[9] M. Chevalier-Boisvert, F. Golemo, Y. Cao, B. Mehta, and L. Paull, "Duckietown environments for openai gym." `https://github.com/duckietown/gym-duckietown`, 2018.

[10] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.