DENSE VS FOCUSED ON DQN FOR REINFORCEMENT LEARNING

Object Design

Document Version

1.0

15.05.2020

Ersin Çebi

Prepared for
ENGR4901

IŞIK UNIVERSITY
COMPUTER
SCIENCE AND
ENGINEERING

Table of Contents

OBJECT DESIGN DOCUMENT

1. Introduction

The purpose of this system is to design a reinforcement learning application and compare Dense and Focusing layers efficiency.This Object Design Document (ODD) defines the object-level design of the app to be developed. Proposed in Python data analysis development strategy.

1.1 Object Design Trade-offs

In this project we will use DQN with dense layers and compare dense loss and accuracy with focusing neuron, this DQN will be applied to OpenAI Gym three environments. Reinforcement learning is an unsupervised learning technique, that is why first we should create data to process, this data is unlabeled and we come up with this data according to the conditions of our environments. An environment limits the agent's actions according to the problem, this problem might be driving a car, a walking person, etc. We could create our own environment by using different libraries but there is OpenAI Gym. In deep q-learning, each level learns to transform its input data into a slightly more abstract and composite representation. In an agent training application, the raw input may be a matrix of actions; the first representational layer may abstract the action and encode states; the second layer may compose and encode arrangements of states; the third layer may encode an action on a state, and the fourth layer may predict a proper outcome that the action of the current state.

1.2 Interface Documentation Guidelines

We will use collab to demonstrate experiments. The system user interface will be made Google Colab .ipynb application pages. In .ipnyb page users can see training and test results, variable changes and model settings.

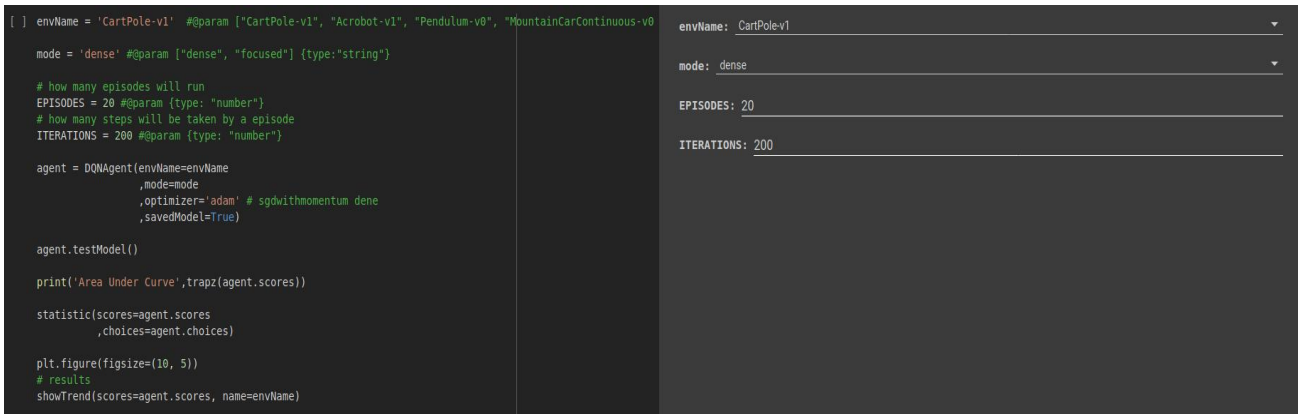

Figure 1.2-1:Colab form interface

1.3 Definitions, Acronyms, and Abbreviations

Focused: Focus scales weights and changes the variance of propagated signals.
Python: Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.
Keras: Kerasis a Python-friendly open-source library for numerical computation that makes machine learning faster and easier.
Colab: Google Colab is a free cloud service and now it supports free GPU! You can; improve
your Python programming language coding skills. develop deep learning applications using
popular libraries such as Keras, TensorFlow, PyTorch, and OpenCV.
DENSE: A dense layer is just a regular layer of neurons in a neural network. Each neuron receives input from all the neurons in the previous layer, thus densely connected.
Q-learning: A model-free reinforcement learning algorithm to learn a policy telling an agent what action to take under what circumstances.
DQN: The DeepMind system used a deep convolutional neural network.

2. Packages

The application is designed and implemented as an Python application. There are a lots of data analysis tool.

Our project classes include Python coding, numpy, OpenAi Gym, matplotlib, FocusingNN, tensorflow, google colab. Tensorflow is used for creating and designing the model part of the application. Python(with jupyter notebook and google colab) is used for code base and activities are used for developing the “UI”, sides of this application. Numpy, matplotlib libraries are used on the application development and visualization of results. The focusingNN is the base of the thesis, which is imported in the project, we change the dense layer with focusing neuron.

3. Class Interfaces

class **DQNAgent**(builtins.object)

DQNAgent(envName, mode, optimizer, seed=42, N=None, savedModel=None)

Methods defined here:

__init__(self, envName, mode, optimizer, seed=42, N=None, savedModel=None)

Initialize self. See help(type(self)) for accurate signature.

clipcallBack(self, varname, clips)

epsilon_greedy_policy(self, state)

get_action(self, action)

isList(self, obj)

play_one_step(self, state)

sample_experiences(self)

saveModel(self)

setAxis(self, state)

testModel(self)

train(self)

training_step(self)

Data descriptors defined here:

__dict__

dictionary for instance variables (if defined)

__weakref__

list of weak references to the object (if defined)

4. References

1. Bruegge B. & Dutoit A.H.. (2010). Object-Oriented Software Engineering Using UML, Patterns, and Java, Prentice Hall, 3rd ed.
2. Géron, Aurélien. Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. " O'Reilly Media, Inc.", 2017.
3. Online: https://github.com/btekgit/FocusingNeuron, date accessed: Feb 2020
4. Boray, F. “An Adaptive Locally Connected Neuron Model: Focusing Neuron.” ArXiv.org, 31 July 2019, https://arxiv.org/abs/1809.09533.