MinerAI environment installation

This guideline is provided for users to install Python environment with libraries used at server, facilitating the running of the test code.

The environment can be installed in two ways.

- Install the environment directly on your PC:
  - Advantages: easy and familiar to those who have worked with Python
  - Disadvantages: there may be difference to the actual running environment due to different OS (the actual environment in which the server runs your code is Ubuntu Server 18.04)
- Using Docker to install:
  - Advantages: the code environment is similar to the actual running environment
  - Disadvantages: installing Docker may be difficult for some older OS

Installation instructions:

- Python 3.6.9 (Ubuntu) – Python 3.7.4 (windows)
- Tensorflow 1.14.0
- Keras 2.3.1
- Numpy 1.18.4
- Pandas 0.15
- PyTorch 1.5.0

1. Installing directly

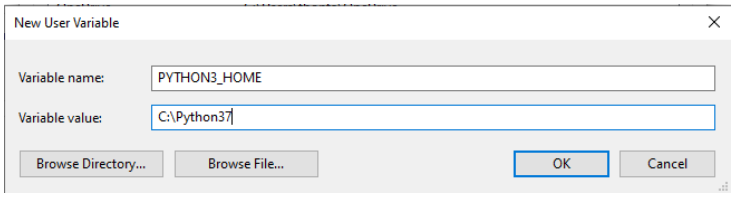
a. Windows

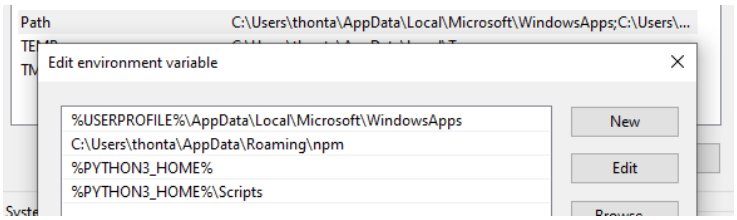
i. Install Python 3.7.4:

Download the executable installer relevant to the OS on your PC at: https://www.python.org/downloads/release/python-374/

(The download link is located in the Files section at the bottom of the page)

Install and set Windows environment variables to PYTHON_HOME /, PYTHON_HOME / Scripts





Verify the installation, make sure the version is 3.7.4

```
D:\>python37 -V
Python 3.7.4
```

ii. Install pip3

Run the following command to install: ***python37 -m pip install --upgrade pip***

Verify the installation, make sure pip3 is installed in Python37

```
D:\>which pip3
/cygdrive/c/Python37/Scripts/pip3
```

iii. Install virtualenv

Run the following command to install: ***pip3 install virtualenv***

Verify the installation, make sure virtualenv is installed in Python37

```
D:\>which virtualenv
/cygdrive/c/Python37/Scripts/virtualenv
```

iv. Install libraries

In order not to affect the normal Python environment, the installation will be done in a virtual environment.

- Change the current directory to the directory you want to install and create a virtual environment:

  virtualenv -p python37 {env_name}

  {env_name}: an environment's name of your choice. For example, if you want to name the environment as Miner, the installation command will be ***virtualenv -p python37 miner***

- Activate the virtual environment
  .\{env_name}\Scripts\activate
- Install libraries:
  pip3 install numpy==1.18.4
  pip3 install keras==2.3.1
  pip3 install pandas==1.0.4
  pip3 install tensorflow==1.14.0

pip3 install torch==1.5.0+cpu torchvision==0.6.0+cpu -f https://download.pytorch.org/whl/torch_stable.html

- Run code: the code is run in the virtual environment set up in the previous step, therefore, make sure to activate the virtual environment in advance.

b. Ubuntu 18.04

i. Install Python 3.6.9

Installation commands:
apt-get update
apt-get install python3
(See here on how to install in other OS versions: https://askubuntu.com/questions/865554/how-do-i-install-python-3-6-using-apt-get)
Verify if the right version is installed:

```
root@8ab45972b8ea:/# python3 -V
Python 3.6.9
```

ii. Install pip

Run the following command to install: ***python3 -m pip install --upgrade pip***
Or: ***sudo apt install python3-pip***

iii. Install virtualenv

Run the following command to install: ***pip3 install virtualenv***

iv. Install libraries

In order not to affect the normal Python environment, the installation will be done in a virtual environment.

- Change the current directory to the directory you want to install and create a virtual environment:

virtualenv -p python3 {env_name}

{env_name}: an environment's name of your choice For example, if you want to name the environment as Miner, the installation command will be ***virtualenv -p python3 miner***

- Activate the virtual environment
cd {env_name}/bin
source ./activate
- Install libraries:
pip3 install numpy==1.18.4
pip3 install keras==2.3.1
pip3 install pandas==0.15
pip3 install tensorflow==1.14.0

pip3 install torch==1.5.0+cpu torchvision==0.6.0+cpu -f https://download.pytorch.org/whl/torch_stable.html

- Run code: the code is run in the virtual environment set up in the previous step, therefore, make sure to activate the virtual environment in advance.

2. Using Docker

We will provide a Docker image which has installed all the environments similar to the actual server environment.

In this section, we will show you how to install the Docker, and how to use the image we provide.

a. Install Docker

You can easily look for comprehensive and detailed instructions to install Docker on the internet. Below are just some examples.

i. Windows:

- Windows 10: Visit the link below to download the installer and install on your PC. https://hub.docker.com/editions/community/docker-ce-desktop-windows/
- Windows 7: Follow the instructions at: https://webme.ie/how-to-install-docker-on-windows-7/

ii. Ubuntu:

- Ubuntu 18.04: Follow the instructions at: https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-18-04
- Ubuntu 16.04: Follow the instructions at: https://docs.docker.com/engine/install/ubuntu/

b. Use Doker image

i. Pull image:

Execute the following command to pull image: ***docker pull codelearnio/miner-ai:training***

Verify if the image has been successfully pulled by the following command: ***docker images –a***

The displayed information will include the Docker image with the following details:

```
D:\>docker images -a
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
codelearnio/miner-ai        training            b71e2ea7dec6        12 minutes ago      2.45GB
```

ii. Use image

This section will instruct you to use some basic commands in Docker container.

For other commands, refer to the following link:

https://docs.docker.com/engine/reference/commandline/docker/

1. Create and run the Docker container from the existing image:

   Create and run the Docker container with the provided Docker image by the following command: ***docker run -it -v {WORKING_DIR}:/v b71e2ea7dec6***

Note: {WORKING_DIR} is the path to the directory where your source is located; For example: if you put the source code in **D: \ MinerAI** directory, the run command will be: ***docker run -it -v D:\MinerAI:/v b71e2ea7dec6***
You can name the container by adding parameters: --name = {name}
Change the current directory to the binded directory: ***cd /v***
Check the files binded to the container: ***ls***

```
D:\>docker run -it -v D:\projects\GameAI\Miner-Training-Local-CodeSample:/v --name=miner b71e2ea7dec6
root@f39ab8375c62:/# cd /v
root@f39ab8375c62:/v# ls
DQNModel.py  GameSocketDummy.py  Memory.py    MinerState.py  TrainingClient.py  bot2.py
Data         Maps                MinerEnv.py  TrainedModels  bot1.py            bot3.py
root@f39ab8375c62:/v#
```

Then execute run ***python3*** command with your source code without any additional installation.
For example: ***python3 TrainingClient.pyt***

2. Check the existing containers: ***docker container ls -a***

```
D:\>docker container ls -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
2f3d9797c028        b71e2ea7dec6        "/bin/bash"         6 seconds ago       Up 6 seconds                                    modest_sutherland
f39ab8375c62        b71e2ea7dec6        "/bin/bash"         2 minutes ago       Exited (0) 16 seconds ago                       miner
```

As shown above, there are 2 containers initialized from the image ***b71e2ea7dec6***: container **2f3d9797c028** is up running while container **f39ab8375c62** has stopped

3. Attach to a running Docker: ***docker attach {container_id}***

```
D:\>docker attach 2f3d9797c028
root@2f3d9797c028:/#
```

Note: You can replace {container_id} with {container_name}

4. Start a stopped Docker container: ***docker start –a {container_id}***

```
D:\>docker start -a miner
root@f39ab8375c62:/#
```

5. Stop a running container: ***docker stop {container_id}***

6. Remove a Docker container: ***docker rm {container_id}***