MSiA 490 Lab 2

Fall 2023

Shuyang Wang

Agenda

- Announcement:
  - Assignment 1 will be released on 10/9 (Monday), due on 10/19 (Thursday).
  - Lab next week covers relevant materials.
  - Strongly recommend you starting early on the assignment.
- Linux Basics
- Open AI Gym
- Tensorflow Keras

Linux basics

- Login your account: ssh
- Basic commands: cd, pwd, ls, cp, mv, rm, mkdir, touch, stat, echo, vim
- Transfer files: sftp, scp
- Run a program: nohup, python -u
- Check the process: top, kill, ps
- GPU usage: export CUDA_VISIBLE_DEVICES=0, nvidia-smi

Example:

```
nohup python -u ./program.py > logs.log 2>&1 & echo $! > job_pid.txt
```

Setup environment on deepdish server

- ssh to your deepdish account

  ```
  ssh <NetID>@mlds-deepdish3.ads.northwestern.edu
  ```

- Create an Anaconda environment

  ```
  conda create -n mldsrl python=3.8
  ```

- Activate the conda environment

  ```
  conda activate mldsrl
  ```

*you can also use other package management tools

Setup environment on deepdish server

- Install pip

  ```
  conda install pip
  ```

- Install tensorflow

  ```
  python -m pip install tensorflow-gpu==2.10.0
  ```

  ```
  python -m pip install typing-extensions==4.3.0
  ```

- Install Open AI Gym

  ```
  python -m pip install "gym[atari, accept-rom-license]"
  ```

*You can also use pytorch or other tf versions.

Verify the packages are installed correctly

Type python in your terminal

```
>>> import gym
>>> env = gym.make('CartPole-v0')
>>> env = gym.make('Pong-v0')

>>> import tensorflow as tf
>>> tf.config.list_physical_devices('GPU')
```

Open AI Gym

Four key functions

```
import gym

1. env = gym.make('CartPole-v0')
2. obs, info = env.reset(seed=42)
3. obs, reward, terminated, truncated, info = env.step(action)
4. env.close()
```

https://www.gymlibrary.dev/content/basic_usage/

```python
import gym
env = gym.make("Pong-v0")


# Roll out 1000 episdoes
for episode in range(1000):
    # Initiate one episode
    observation, info = env.reset(seed=42)

    obs_history = []
    reward_history = []
    action_history = []

    terminated, truncated = False, False

    # Roll out one episode
    while not terminated:
        action = env.action_space.sample() # Use your policy here
        observation, reward, terminated, truncated, info = env.step(action)

        obs_history.append(observation)
        reward_history.append(reward)
        action_history.append(action)

    # TODO: Update your policy using the collected episodes


env.close()
```

Tensorflow Keras

Verify that tensorflow 2 is installed and can access GPU

- `import tensorflow as tf`
- `tf.__version__`
- `tf.config.list_physical_devices('GPU')`

```
import tensorflow as tf
import os
os.environ["CUDA_VISIBLE_DEVICES"] = "0"
import warnings
warnings.filterwarnings('ignore')
```

```python
## Load Dataset
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

# Build a Neural Network

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10)
])

# Define the loss function
loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)


# Compile the model, specify optimizer, loss function, metrics for evaluation
model.compile(optimizer='adam',
              loss=loss_fn,
              metrics=['accuracy'])

# Train the model
model.fit(x_train, y_train, epochs=20)

# Evaluate the model
model.evaluate(x_test, y_test, verbose=2)
```