How to efficiently makes a program

DGIST

Dongsun Kim

10/May/2019

UGRP Meeting

Error

- **Your best friends**
  - Warning and error is one of the most frequently encounter
  - Types of error – **infinite**
  - General cases – pioneers already discover that error
    1. Read error → **Google**
    2. Read code → **Google**
  - Special cases – you have to deal with by yourselves
    0. At least try to deal with the error
    1. Too simple that nobody explain about it
    2. Stupid mistakes, e.g., did not add lib folder
    3. **Don't know how to ask to the Google**

    → **debugger**

Just put enough time will solve the problem

How to use debugger for efficient programming – first step

Coding

- **What you have to do until 5.17**
  - Design ultra dense network environment
  - Determine which reinforcement learning algorithm to use

    Should be done
  - **Do not reinvent the wheel**
    1. Google is your another best friend
    2. Github has plenty of good open source materials
    3. The author's homepage might gives the source code
  - Run the sample code
    0. **Understand the code**
    1. Run the original code
    2. Makes a **pseudo code**
    3. Remakes a simplest code based on the original code
    4. Makes **"Remark"** for better understanding of code
  - Makes advanced code

Run the Code

- **Run the original code**
  - Recommend to use **ssh** – secure and fast
  - Download **github code** 레포지토리
    - git clone <address>
    - e.g., git clone https://github.com/PacktPublishing/Deep-Reinforcement-Learning-Hands-On/
  - Run the code (recommend to use terminal not notebook)

```
[dongsun@whoami Chapter09]$ python 01_cartpole_dqn.py
DQN(
  (net): Sequential(
    (0): Linear(in_features=4, out_features=128, bias=True)
    (1): ReLU()
    (2): Linear(in_features=128, out_features=2, bias=True)
  )
)
19: reward:  17.00, mean_100:  17.00, epsilon: 1.00, episodes: 1
49: reward:  29.00, mean_100:  23.00, epsilon: 0.99, episodes: 2
67: reward:  17.00, mean_100:  21.00, epsilon: 0.99, episodes: 3
84: reward:  16.00, mean_100:  19.75, epsilon: 0.98, episodes: 4
102: reward:  17.00, mean_100:  19.20, epsilon: 0.98, episodes: 5
120: reward:  17.00, mean_100:  18.83, epsilon: 0.98, episodes: 6
145: reward:  24.00, mean_100:  19.57, epsilon: 0.97, episodes: 7
176: reward:  30.00, mean_100:  20.88, epsilon: 0.96, episodes: 8
```

Error – Missed Library

- Q.

  1. **Makes your own environment (recommend)**

  – https://stackoverflow.com/questions/7465445/how-to-install-python-modules-without-root-access

  2. Ask me to add libraries

  – 기왕 할거면 한꺼번에.. 그리고 미리미리

Error – Basic Grammar

- Main idea : learn from basic
  - Version check – ‘pip list | grep <lib>’
  - Search the tutorial for the version
    - e.g., Python 3.7.3 tutorial, PyTorch tutorial, Tensorflow tutorial
  - Github – search the library

- Commonly used library
  - **numpy**, scipy
  - matplotlib
  - **gym**
  - Among **pytorch**, tensorflow, keras, …
  - ptan
  - mujoco (unless necessary do not use)
  - pdb, or **ipdb**

Error – Unknown

- Type error
  - python, numpy, or pytorch types error
  - Assignment of different types or **shape** incur errors
  - Check error for short code is easy → read carefully
  - Check error for long code is hard → use python debugger (or c debugger)

- ipdb – ipython interface like python debugger
  - Add following line before the error
  - **'import ipdb; ipdb.set_trace()'**
  - Set the breakpoint for the code

Coding-Example

- Chaper09 of Deep~ github code

```python
#!/usr/bin/env python3
import gym
#import myenv
import ptan
import numpy as np
from tensorboardX import SummaryWriter

import torch
import torch.nn as nn
import torch.optim as optim

GAMMA = 0.99
LEARNING_RATE = 0.01
BATCH_SIZE = 8

EPSILON_START = 1.0
EPSILON_STOP = 0.02
EPSILON_STEPS = 5000

REPLAY_BUFFER = 50000
```

Import library

Set values for algorithm

Coding-Example

- Chaper09 of Deep~ github code

```python
class DQN(nn.Module):
    def __init__(self, input_size, n_actions):
        super(DQN, self).__init__()

        self.net = nn.Sequential(
            nn.Linear(input_size, 128),
            nn.ReLU(),
            nn.Linear(128, n_actions)
        )

    def forward(self, x):
        return self.net(x)


def calc_target(net, local_reward, next_state):
    if next_state is None:
        return local_reward
    state_v = torch.tensor([next_state], dtype=torch.float32)

    next_q_v = net(state_v)
    best_q = next_q_v.max(dim=1)[0].item()
    return local_reward + GAMMA * best_q
```

Define Neural Network

Define value iteration

Coding-Example

- Chaper09 of Deep~ github code

```python
if __name__ == "__main__":
    env = gym.make("CartPole-v0")
    #env = gym.make("UAVHC-v1")
    writer = SummaryWriter(comment="-cartpole-dqn")

    net = DQN(env.observation_space.shape[0], env.action_space.n)
    print(net)

    selector = ptan.actions.EpsilonGreedyActionSelector(epsilon=EPSILON_START)
    agent = ptan.agent.DQNAgent(net, selector, preprocessor=ptan.agent.float32_preprocessor)
    exp_source = ptan.experience.ExperienceSourceFirstLast(env, agent, gamma=GAMMA)
    replay_buffer = ptan.experience.ExperienceReplayBuffer(exp_source, REPLAY_BUFFER)

    optimizer = optim.Adam(net.parameters(), lr=LEARNING_RATE)
    mse_loss = nn.MSELoss()

    total_rewards = []
    step_idx = 0
    done_episodes = 0

    while True:
```

set gym environment
set neural network

Use ptan wrapper

Set optimizer for training
Initialize training pipeline

Q. What is ptan and how should we deal it?

```python
while True:
    step_idx += 1
    selector.epsilon = max(EPSILON_STOP, EPSILON_START - step_idx / EPSILON_STEPS)
    replay_buffer.populate(1)

    if len(replay_buffer) < BATCH_SIZE:
        continue

    # sample batch
    batch = replay_buffer.sample(BATCH_SIZE)
    batch_states = [exp.state for exp in batch]
    batch_actions = [exp.action for exp in batch]
    batch_targets = [calc_target(net, exp.reward, exp.last_state)
                     for exp in batch]
    # train
    optimizer.zero_grad()
    states_v = torch.FloatTensor(batch_states)
    net_q_v = net(states_v)
    target_q = net_q_v.data.numpy().copy()
    target_q[range(BATCH_SIZE), batch_actions] = batch_targets
    target_q_v = torch.tensor(target_q)
    loss_v = mse_loss(net_q_v, target_q_v)
    loss_v.backward()
    optimizer.step()

    # handle new rewards
    new_rewards = exp_source.pop_total_rewards()
    if new_rewards:
        done_episodes += 1
        reward = new_rewards[0]
        total_rewards.append(reward)
        mean_rewards = float(np.mean(total_rewards[-100:]))
        print("%d: reward: %6.2f, mean_100: %6.2f, epsilon: %.2f, episodes: %d" % (
            step_idx, reward, mean_rewards, selector.epsilon, done_episodes))
        writer.add_scalar("reward", reward, step_idx)
        writer.add_scalar("reward_100", mean_rewards, step_idx)
        writer.add_scalar("epsilon", selector.epsilon, step_idx)
        writer.add_scalar("episodes", done_episodes, step_idx)
        if mean_rewards > 195:
            print("Solved in %d steps and %d episodes!" % (step_idx, done_episodes))
            break
writer.close()
```

Epsilon greedy

Set batch

Initialize optimizing step
Get value from network
Get error in value function
Update network based on loss

Logging performance

Question

- Pseudo code - practice

- What is the intention of the code?
  - Algorithm
  - Environment
  - Hyperparameters
  - Networks
  - Why they save?

Error – Unknown

- ipdb – ipython interface like python debugger

```
Traceback (most recent call last):
  File "01_cartpole_dqn.py", line 80, in <module>
    for exp in batch]
  File "01_cartpole_dqn.py", line 80, in <listcomp>
    for exp in batch]
  File "01_cartpole_dqn.py", line 42, in calc_target
    next_q_v = net(state_v)
  File "/usr/lib/python3.7/site-packages/torch/nn/modules/module.py", line 489, in __call__
    result = self.forward(*input, **kwargs)
  File "01_cartpole_dqn.py", line 34, in forward
    return self.net(x)
  File "/usr/lib/python3.7/site-packages/torch/nn/modules/module.py", line 489, in __call__
    result = self.forward(*input, **kwargs)
  File "/usr/lib/python3.7/site-packages/torch/nn/modules/container.py", line 92, in forward
    input = module(input)
  File "/usr/lib/python3.7/site-packages/torch/nn/modules/module.py", line 489, in __call__
    result = self.forward(*input, **kwargs)
  File "/usr/lib/python3.7/site-packages/torch/nn/modules/linear.py", line 67, in forward
    return F.linear(input, self.weight, self.bias)
  File "/usr/lib/python3.7/site-packages/torch/nn/functional.py", line 1352, in linear
    ret = torch.addmm(torch.jit._unwrap_optional(bias), input, weight.t())
RuntimeError: size mismatch, m1: [2 x 2], m2: [4 x 128] at /build/python-pytorch/src/pytorch-1.
0.1-cuda/aten/src/TH/generic/THTensorMath.cpp:940
[dongsun@whoami Chapter09]$
```

Error – Unknown

- ipdb – ipython interface like python debugger
  - https://pythonadventures.wordpress.com/tag/ipdb/
  - and more…

```
37 def calc_target(net, local_reward, next_state):
38     if next_state is None:
39         return local_reward
40     state_v = torch.tensor([next_state], dtype=torch.float32)
41     state_v = state_v.reshape(-1, 2)
42     import ipdb; ipdb.set_trace()  # XXX BREAKPOINT
43     next_q_v = net(state_v)
44     best_q = next_q_v.max(dim=1)[0].item()
45     return local_reward + GAMMA * best_q
```

Error – Unknown

- ipdb – ipython interface like python debugger

```
[dongsun@whoami Chapter09]$ python 01_cartpole_dqn.py
DQN(
  (net): Sequential(
    (0): Linear(in_features=4, out_features=128, bias=True)
    (1): ReLU()
    (2): Linear(in_features=128, out_features=2, bias=True)
  )
)
> /home/dongsun/Personal/Code/RL/tutorial/2_Deep-Reinforcement-Learning-Hands-On/Chapter09/01_c
artpole_dqn.py(43)calc_target()
     42     import ipdb; ipdb.set_trace()  # XXX BREAKPOINT
---> 43     next_q_v = net(state_v)
     44     best_q = next_q_v.max(dim=1)[0].item()

ipdb> state_v.shape
torch.Size([2, 2])
ipdb> print(net)
DQN(
  (net): Sequential(
    (0): Linear(in_features=4, out_features=128, bias=True)
    (1): ReLU()
    (2): Linear(in_features=128, out_features=2, bias=True)
  )
)
ipdb> tmp_state=state_v.reshape(-1, 4)
ipdb> net(tmp_state)
tensor([[-0.0105,  0.0009]], grad_fn=<AddmmBackward>)
ipdb> work!!
```

Conclusion

- Start point
  - Google, *github, tutorials*
- Practice makes perfect
  - Pseudo code
  - **Read the others codes and mimic**
- Error handling
  - **Most of errors are already experienced and solved by Google**
  - **Knowing how to ask** is important skill to enhance the code
  - **Python debugger** can help unknown error

Any Questions?

THANK YOU