MLDS 2018 Spring HW4-1 - Policy Gradient

2018/6/1

ntu.mldsta@gmail.com

Time Schedule

- June 1st 4-1 announce
  - Policy Gradient
- June 8th 4-2 announce
  - Deep Q learning
- June 15th 4-3 announce
  - Actor-Critic
- July 6th 23:59 Deadline (all in one)

Outline

- **Introduction**
  - Game Playing : Pong
- **Deep Reinforcement Learning**
  - Policy Gradient
  - Improvements to Policy Gradient
- **Training Hints**
- **Grading & Format**
  - Grading Policy
  - Code Format
  - Report
  - Submission

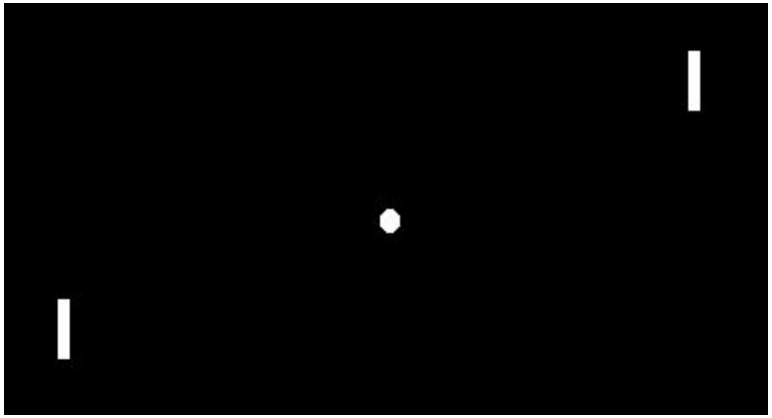
Introduction

Game Playing

- Implement an agent to play Atari games using Deep Reinforcement Learning
- In this homework, you are required to implement **Policy Gradient**
- The Pong environment is used in this homework

Environment

Pong



https://gym.openai.com/envs/

Deep Reinforcement Learning

Policy Gradient

function **REINFORCE**

Initialise θ arbitrarily

for each episode $\{s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**

for $t = 1$ to $T - 1$ **do**

$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$

end for

end for

return θ

end function

s_i : state at time i

a_i : action at time i

r_i : reward by a_i

$\pi_\theta(s, a) = P[a|s, \theta]$: θ is your model parameter

v_t : long-term value at time t

$v(s) = E[\ G_t | s_t = s]$

$G_t = \sum_{k=0}^{\inf} \gamma^k r_{t+k+1}$

- Update per step → SGD → High Variance
- Update per episode or by mini batch
  - episode : A player win the game (21)
  - mini batch : someone get some points

REINFORCE Baseline on Pong

Training loop(simplest version):

a. Play until a game is over(one player gets 21 points) with policy network π_θ and store (s,a,r) tuples into memory m.
b. Discount and normalize rewards in memory into r' to reduce variance
c. Approximate gradient $\nabla_\theta J(\theta) \approx \sum_{(s_t, a_t, r'_t) \in m} \nabla_\theta \log \pi_\theta(a_t|s_t) r'_t$
d. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
e. Clear the memory m

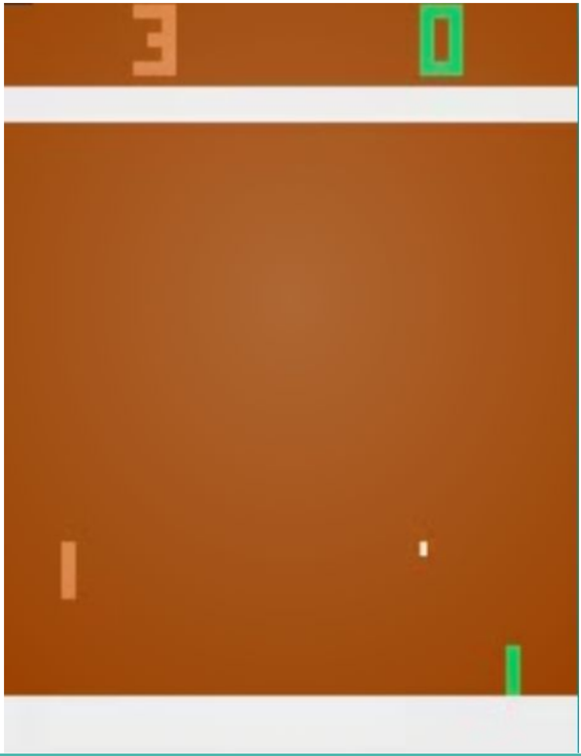
Improvements to Policy Gradient

- Variance Reduction
- Natural Policy Gradient
- Trust Region Policy Optimization
- Proximal Policy Optimization

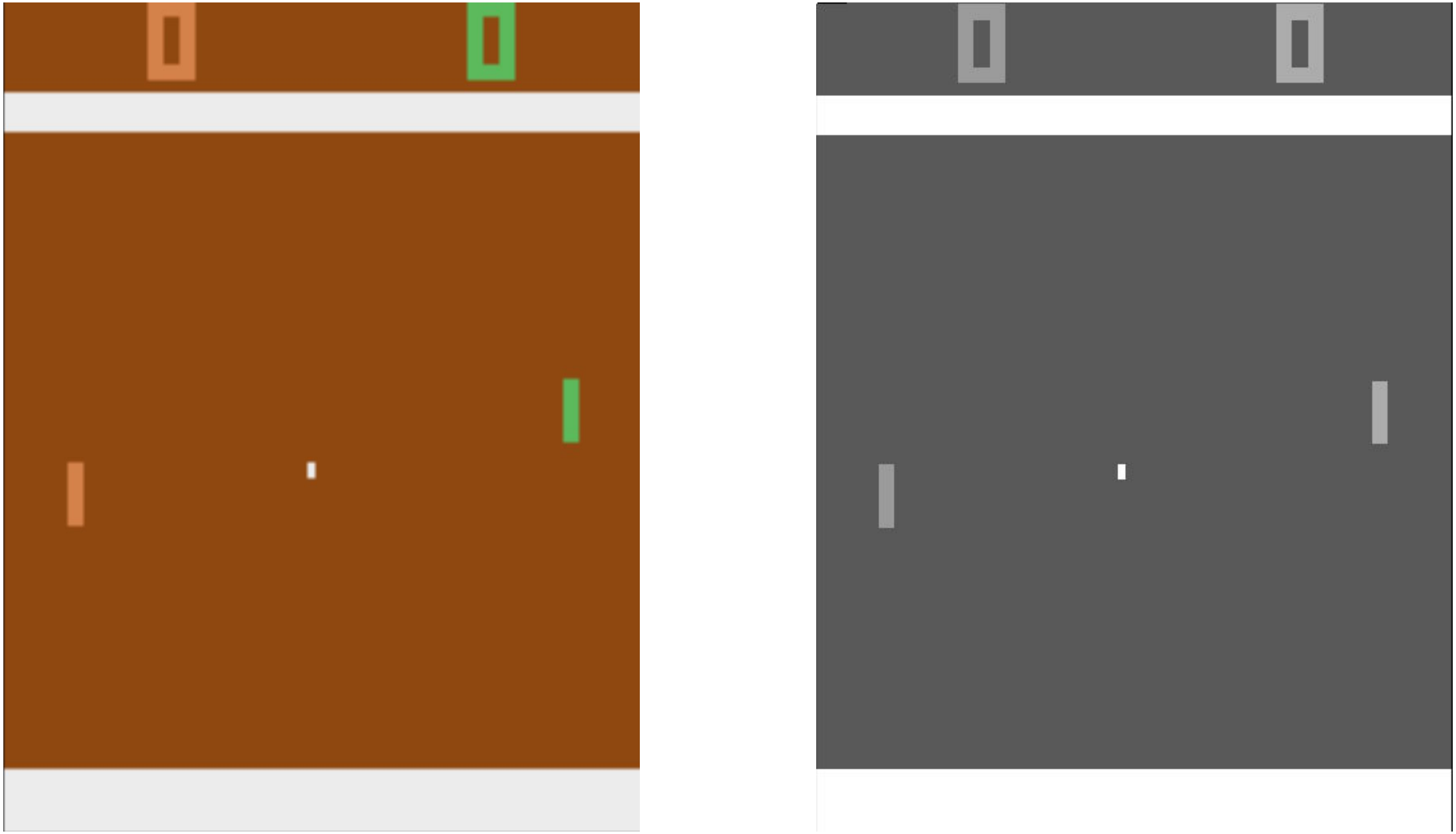
http://rll.berkeley.edu/deeprlcourse/f17docs/lecture_4_policy_gradient.pdf
http://rll.berkeley.edu/deeprlcourse/f17docs/lecture_13_advanced_pg.pdf

Preprocessing for States

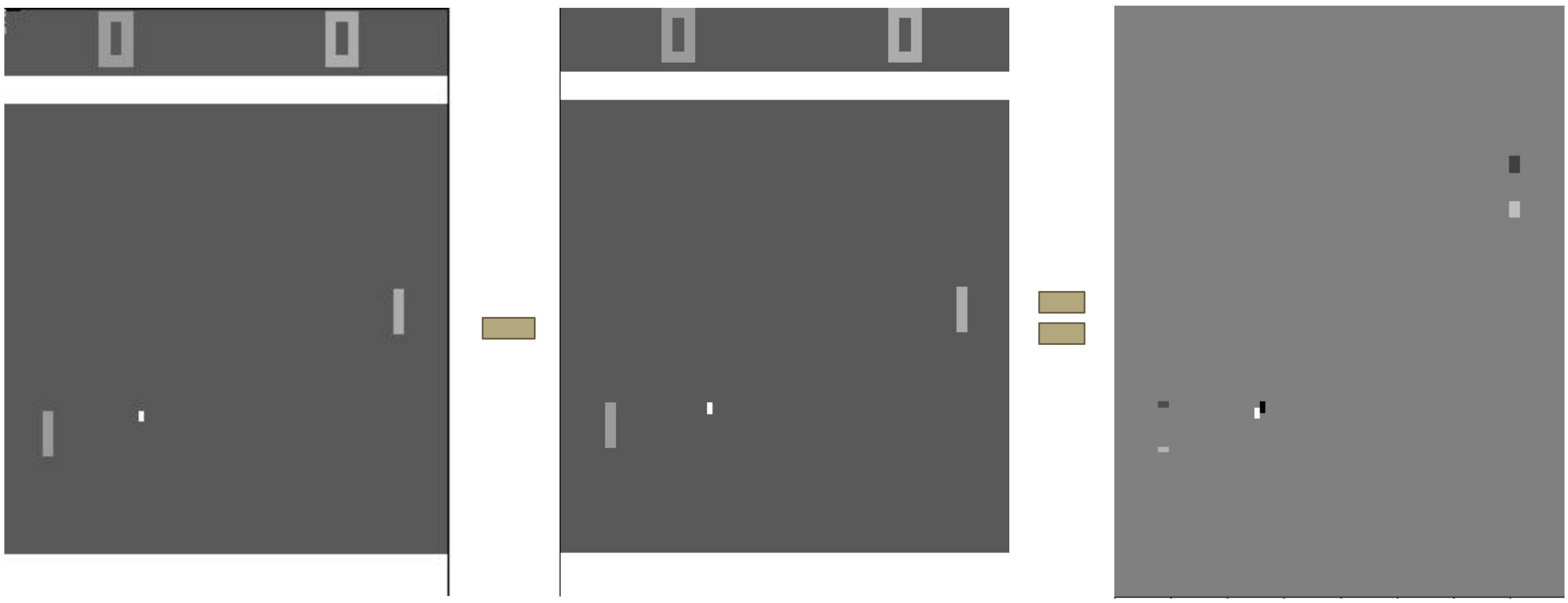
- Which is better ?
  - rgb channel or gray scale
    - 0.2126 * Red + 0.7152 * Green + 0.0722 * Blue
  - single or residual
    - $s'(t) = s(t+1) - s(t)$
    - represent change of pixel
  - scoreboard yes or no?



RGB vs Gray scale



Residual State



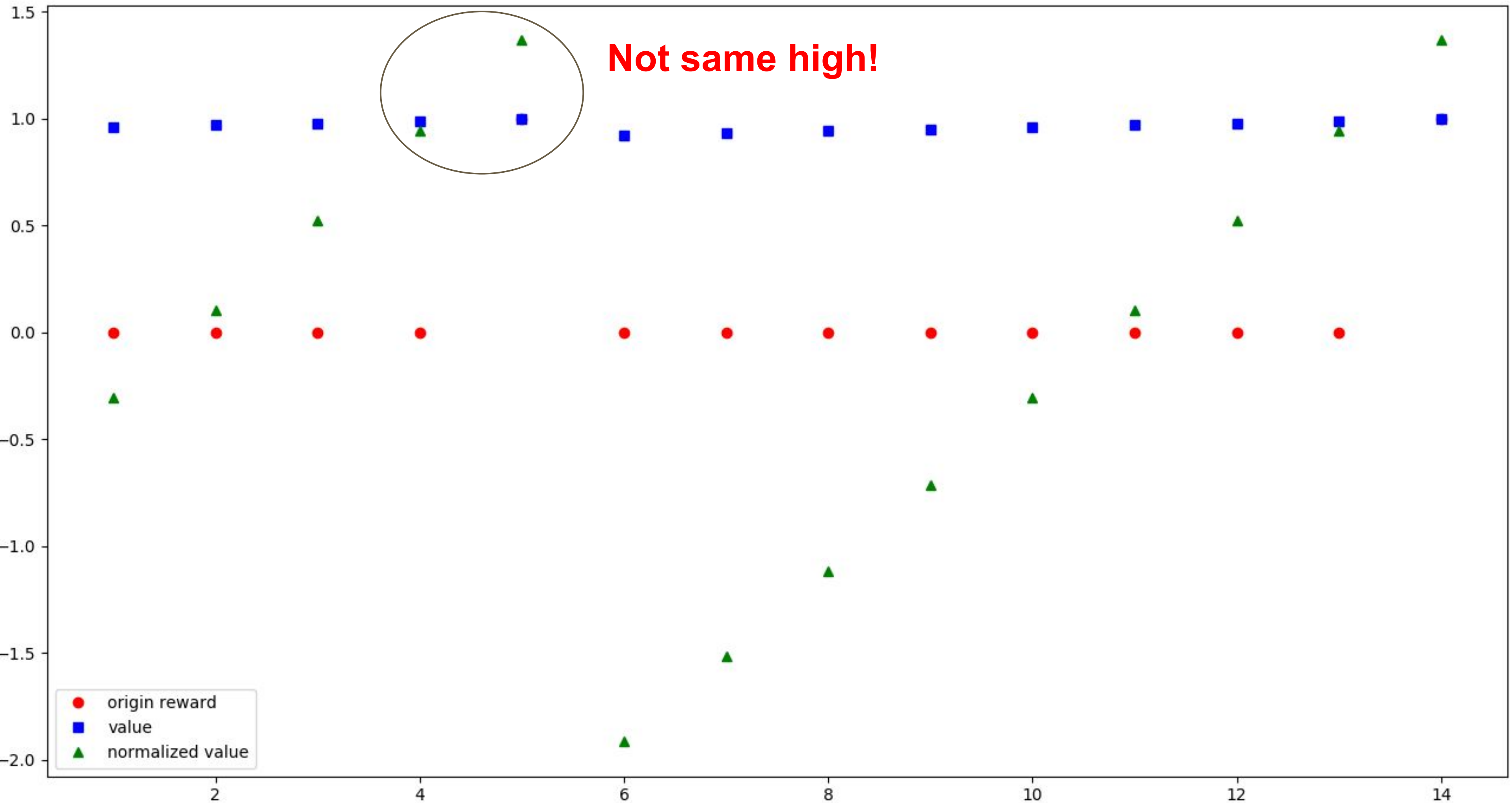
S2 S1 S1’

Training Hint

Reward and Action

- Reward normalization
  - More stable
  - http://karpathy.github.io/2016/05/31/rl
  - https://arxiv.org/pdf/1506.02438.pdf
- Action space reduction
- Reset the running add of discounted reward to zero if a player scores (Pong specific)

Reward normalization

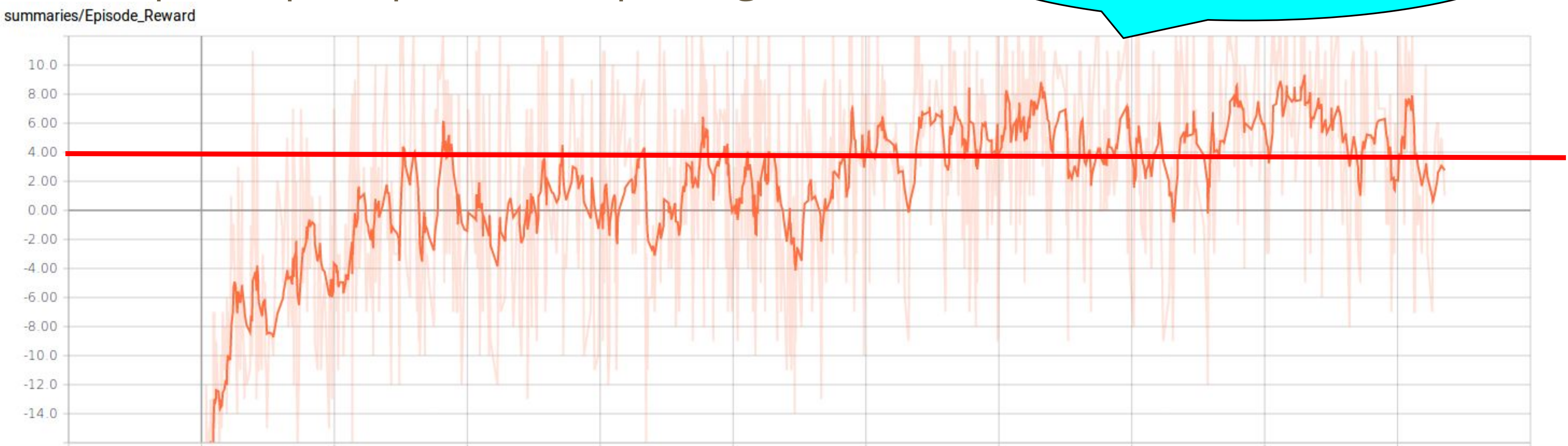


Training Hint

Training Plot

- The unit of x-axis is 1000 episode
- Around 6000 episode to reach baseline in “average”
- Mind your preprocessing if your curve differs from this too much
- Baseline Network Structure : Flatten + Two-layer FNN
  - 256 dimension hidden layer
  - output layer (action space size)
- Update per episode (21 point game)

Freeze random seed!



Grading Policy

- Code Baseline (5%)

- Report (5%)

Grading & Format

Baseline (5%)

- Policy Gradient (5%)
  - Getting averaging reward in 30 episodes over **3** in **Pong**
  - Without OpenAI's Atari wrapper & reward clipping
  - Improvements to Policy Gradient are allowed

Grading & Format

Code Format

- Please download the sample files from github
- Follow the instructions in README to install required packages
- **Four** functions you should implement in agent_pg.py
    1. __init__(self, env, args)
    2. init_game_setting(self)
    3. train(self)
    4. make_action(self, state, test)
- **DO NOT** add any parameter in __init__(), init_game_setting() and make_action()
- You can add new methods in the agent_pg.py

Grading & Format

Report (5%)

- Up to 6 pages (4-1 + 4-2 + 4-3), in Chinese
- Describe your Policy Gradient model (1%)
- Plot the learning curve to show the performance of your Policy Gradient on Pong (1%)
    - X-axis: number of time steps
    - Y-axis: average reward in last 30 episodes
- Implement 1 improvement method on page 8
    - Describe your tips for improvement (1%)
    - Learning curve (1%)
    - Compare to the vallina policy gradient (1%)

Grading & Format

Late submission

- Please fill the late submission form first only if you will submit HW late
- Please push your code before you fill the form
- There will be 25% penalty per day for late submission, so you get 0% after four days
- You get 0% if the required files has bug.
    - If the error is due to the format issue, please come to fix the bug at the announced time, or you will get 10% penalty afterwards.

Grading & Format

Submission

- Deadline: **2018/7/6 23:59 (GMT+8)**
- Your github **MUST** have 5 files under directory hw4/
  - agent_dir/agent_pg.py
  - agent_dir/agent_dqn.py
  - [saved_model_file] * 2
  - report.pdf
  - argument.py (optional)
  - README (optional)
  - download.sh (optional)
  - other files you need
- If your model is too large for github, upload it to a cloud space and write download.sh to download the model
- Do not upload any file named the same with other sample codes

Grading & Format

Grading

- Please use Python with version >= 3.5
- The TAs will execute 'python3 test.py --test_pg --test_dqn' to run your code on **ubuntu**
- The execution for both model should be done within 10 minutes respectively, excluding model download
- Allowed packages
    - PyTorch v0.3.0
    - Tensorflow r1.6 (CUDA 9.0)
    - Numpy
    - Scipy
    - Pandas
    - Python Standard Lib
- No keras !!!! No keras !!!! No keras !!!! No keras !!!! No keras !!!!
- If you use other packages, please ask for permission first

Related Materials

- Course & Tutorial:
  - Berkeley Deep Reinforcement Learning, Fall 2017
  - David Silver RL course
  - Nips 2016 RL tutorial
- Blog:
  - Andrej Karpathy's blog
  - Arthur Juliani's Blog
  - Openai
- Text Book:
  - Reinforcement Learning: An Introduction
- Repo:
  - https://github.com/williamFalcon/DeepRLHacks