Deep Learning Competition 04: Play Flappy Bird with Policy Gradient

Pin-Yu Wang & DataLab

Outline

- Play Flappy Bird with Policy Gradient
- Evaluation
- Timeline & Scoring
- Hints

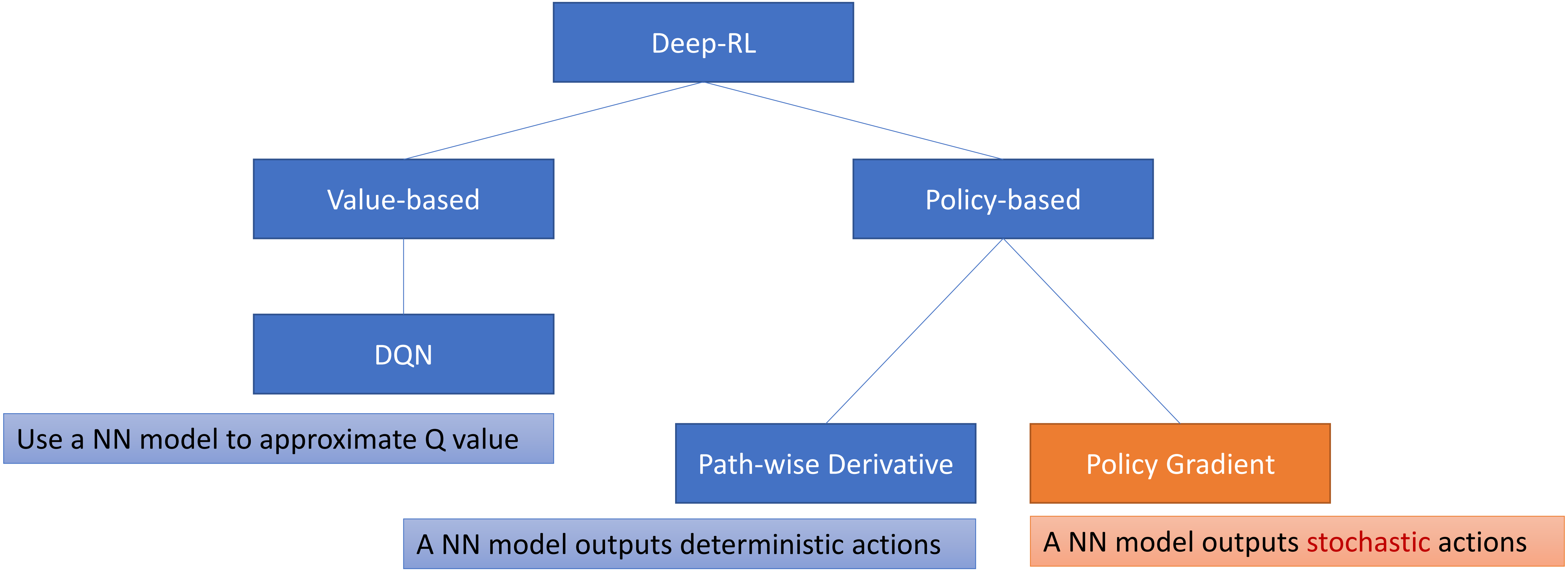
Outline

- Play Flappy Bird with Policy Gradient
- Evaluation
- Timeline & Scoring
- Hints

Play Flappy Bird with Policy Gradient

- Train a RL agent to play flappy bird with policy gradient

Policy Gradients



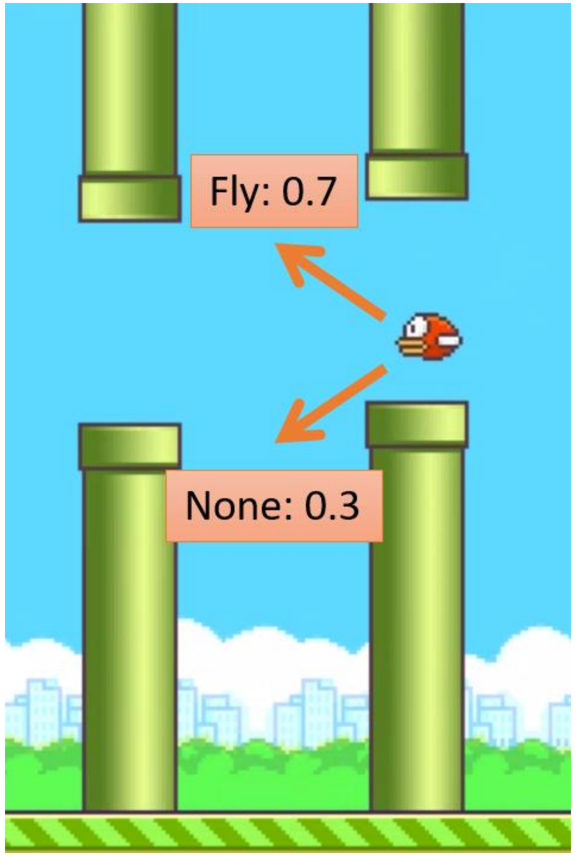
Vanilla Policy Gradient - REINFORCE

REINFORCE (MC estimate): initialize Φ arbitrarily, iterate until converge:

1. Run episodes $\{\tau^{(i)}\}_i$ by sampling actions from $g(\cdot;\Phi)$
2. For each time step t in an episode, compute $R^{(i,t)} = \sum_{t'=t}^{H^{(i)}} \gamma^{t'} R^{(i,t')}$
3. Update Φ using SGD:

$$\Phi \leftarrow \Phi + \eta \nabla_\Phi \hat{J}, \text{ where}$$

$$\nabla_\Phi \hat{J}(\Phi) = \sum_{i,t} \nabla_\Phi \log \mathrm{P}(\boldsymbol{a}^{(i,t)} \mid \boldsymbol{s}^{(i,t)}; \Phi) R^{(i,t)}.$$



REINFORCE + baseline

- MC: High Variance
- How to lower the variance of vanilla policy gradient algorithm?
  - subtract a baseline which is independent with actions (V value)

$$\nabla_\Phi J(\Phi) \propto \sum_{i,t} \nabla_\Phi \log \mathrm{P}(\boldsymbol{a}^{(i,t)} | \boldsymbol{s}^{(i,t)}; \Phi)\left(\sum_{t'=t}^{H^{(i)}} \gamma^{t'} R^{(i,t')} - b\right)$$

$$b = \frac{1}{|\{j,t'' : \boldsymbol{s}^{(j,t'')} = \boldsymbol{s}^{(i,t)}\}|} \sum_{j,t'' : \boldsymbol{s}^{(j,t'')} = \boldsymbol{s}^{(i,t)}} \sum_{t'=t''}^{H^{(j)}} \gamma^{t'} R^{(j,t')}$$

Actor-Critic

- Again! Why not use a DNN to approximate Q or V

  $\sum_{t'=t}^{H^{(i)}} \gamma^{t'} R^{(i,t')} \approx Q_\pi(\boldsymbol{s}^{(i,t)}, \boldsymbol{a}^{(i,t)})$ can be approximated by $R^{(i,t)} + \gamma f_{V_\pi}(\boldsymbol{s}^{(i,t+1)}; \Theta)$
  - No need for f_{Q_π}

- Algorithm (TD): initialize Θ and Φ arbitrarily, iterate until converge:
  1. Take an action $\boldsymbol{a}$ from $\boldsymbol{s}$ using $g(\boldsymbol{s};\Phi)$
  2. Observe $\boldsymbol{s}'$ and reward R, compute $\hat{Q}_\pi \leftarrow R + \gamma f_{V_\pi}(\boldsymbol{s}';\Theta)$
  3. Update f_{V_π}:

     $$\Theta \leftarrow \Theta - \eta \nabla_\Theta \left[\hat{Q}_\pi - f_{V_\pi}(\boldsymbol{s};\Theta)\right]^2$$

  4. Update g_π:

     $$\Phi \leftarrow \Phi + \lambda \nabla_\Phi \log \mathrm{P}(\boldsymbol{a}|\boldsymbol{s};\Phi)(\hat{Q}_\pi - f_{V_\pi}(\boldsymbol{s};\Theta))$$

Outline

- Play Flappy Bird with Policy Gradient
- Evaluation
- Timeline & Scoring
- Hints

Evaluation

- No Kaggle this time, but you can submit your average episode reward to the public sheet.
    - Please use seed 2021 in the test environment.
- TA will run your model 100 times with a private seed to get an average episode reward and rank it.

Model Requirements

- Please save your ENTIRE model via model. Save()
  - Quick information
  - More information

Model Requirements

- Model Input
  - TA will pass relative states into your model
  - The type of states has been converted to a tensor

```
<tf.Tensor: shape=(1, 8), dtype=float32, numpy=
array([[ 256.,    0.,  309., -129.,  -29.,  453., -161.,  -61.]],
      dtype=float32)>
```

- Model Output
  - The output should be the action probability for the given states.
  - The type of the output should be a tensor.

```
tf.Tensor([[0.5384239  0.46157604]], shape=(1, 2), dtype=float32)
```

Example

```
class ExampleModel(tf.keras.Model):
    def __init__(self):
        super().__init__()
        self.dense = layers.Dense()
        self.softmax = layers.Softmax()
                                    Input
    def call(self, relative_state):
        x = self.dense1(inputs=relative_state)
        action_prob = self.softmax(x)
        return action_prob
                  output
```

Example

```
model = ExampleModel()
model.save("./saved_model/policy_gradient")
```

Outline

- Play Flappy Bird with Policy Gradient
- Evaluation
- Timeline & Scoring
- Hints

Competition Timeline

- 2020/12/29 competition announced
- 2020/1/19 (Tue.) competition & report deadline

Scoring

- Ranking on private leaderboard (80%)
  - TA 60
    - Your agent is able to pass 1 pipe in average
    - Average episode reward >= -4
  - TA 80
    - Your agent is able to pass 5 pipes in average
    - Average episode reward >= 0
- Report (20%)
  - Your report should contain following points
    - Models you tried during the competition.
    - What makes your agent work? For example, discount rate, optimizer…
    - Any trick worth mentioning.

Submission

- Submit the link of Google Drive containing two notebooks and an entire model file to ilms.
  - Name the notebook of your training code as
    - DL_comp4_{Your Team number}.ipynb
  - Name the notebook of the testing environment as
    - DL_comp4_{Your Team number}_Test_Environment.ipynb.
  - Name your entire model file as
    - DL_comp4_{Your Team number}_model
    - Please make sure your model works well in the test environment

Precautions

- Run TA's test environment before submitting the model. You will get 0 point if we can't run your model.
- You will get 0 point if not using policy-gradient methods (stochastic policy).
- Plagiarism is not allowed.

Outline

- Play Flappy Bird with Policy Gradient
- Evaluation
- Timeline & Scoring
- Hints

Hints

- It is pretty OK not using @tf.function in this competition.
- Printing out action probabilities for each step is helpful for debugging.
- Don’t forget to play the discount factor, which is a significant parameter in reinforcement learning.
- Preprocessing input states and rewards might be helpful

Reference

Policy-based	
REINFORCE(PG)	Simple statistical gradient-following algorithms for connectionist reinforcement learning. Ronald J. Williams 1992.
Trust Region Policy Optimization (TRPO)	Abbeel et al. Trust region policy optimization. Schulman et al.2015.
Proximal Policy Optimization (PPO)	Proximal policy optimization algorithms. Schulman et al. 2017.

Actor-Critic	
Actor-Critic (AC)	Actor-critic algorithms. Konda er al. 2000.
Asynchronous Advantage Actor-Critic (A3C)	Asynchronous methods for deep reinforcement learning. Mnih et al. 2016.

Reference

Algorithms	Action Space	Policy
DQN (double, dueling, PER)	Discrete Only	--
AC	Discrete/Continuous	Stochastic
PG	Discrete/Continuous	Stochastic
DDPG	Continuous	Deterministic
TD3	Continuous	Deterministic
SAC	Continuous	Stochastic
A3C	Discrete/Continuous	Stochastic
PPO	Discrete/Continuous	Stochastic
DPPO	Discrete/Continuous	Stochastic
TRPO	Discrete/Continuous	Stochastic