Coding Reinforcement Learning Papers

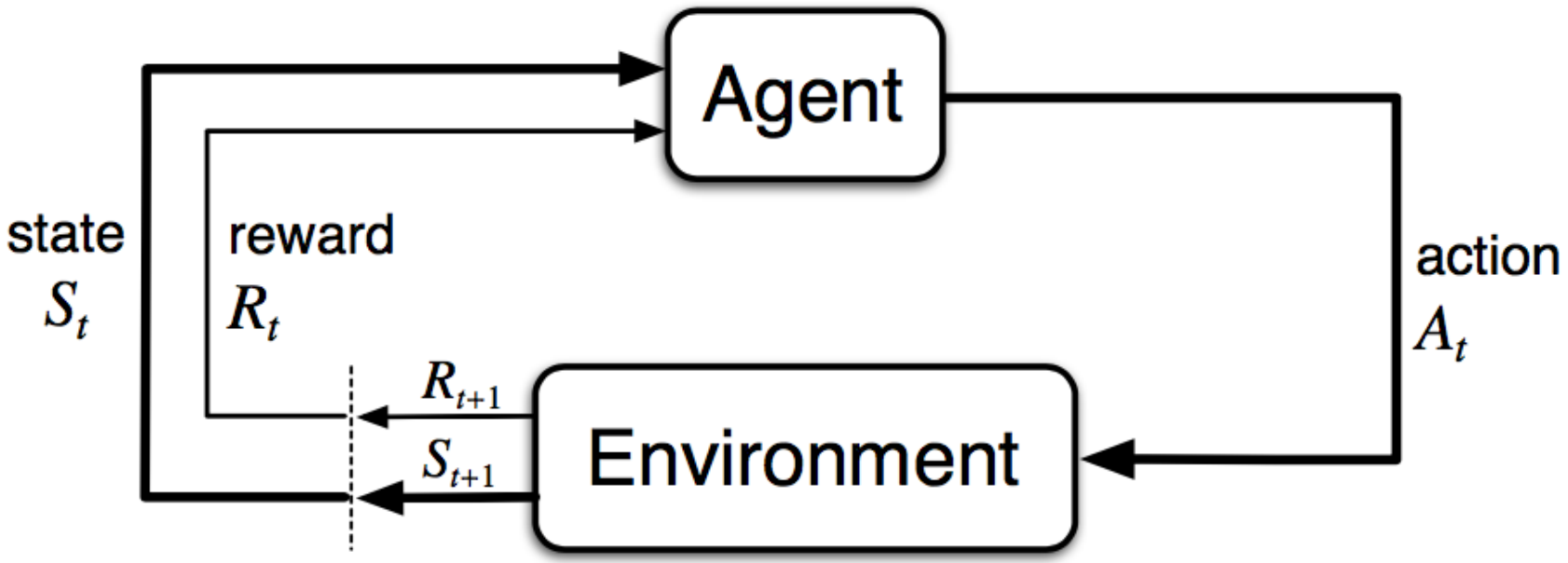
Shangtong Zhang

MLTrain Workshop @ NIPS 2017

December 9, 2017

Outline

Reinforcement Learning



Sutton and Barto (1998)

Reinforcement Learning

- An MDP $< \mathcal{S}, \mathcal{A}, r, p, \gamma >$
- A policy $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$
- State value function

$$v_\pi(s) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s]$$

- State-action value function

$$q_\pi(s, a) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a]$$

- Actor(π)-Critic(v, q)

General Tips

- Exploitation or Exploration
- As Simple As Possible
- Debugging and Tuning
- Framework Selection

General Tips - Exploitation or Exploration

Exploitation:

- missing important details
- difficult to debug

Exploration:

- build our own codebase
- modularize our code

General Tips - As Simple As Possible

- Simple tasks, e.g. Cart Pole & Pendulum
- Simple networks, e.g. single hidden layer
- Less tricks
- . . .

General Tips - Debugging and Tuning

- Visualization
  - gradients, activations, . . .
  - state/reward distribution, q-value
  - episode length, episode return
- Be patient
  - A3C + Atari: about 1 hour
  - DQN + Atari: about 1 day
  - Async Q/Sarsa + Atari: days
- ~~Random seed~~

General Tips - Framework Selection

- PyTorch
- Tensorboard[1]

[1] https://github.com/lanpa/tensorboard-pytorch

Shallow RL

- Tabular Methods
- Linear Function Approximation

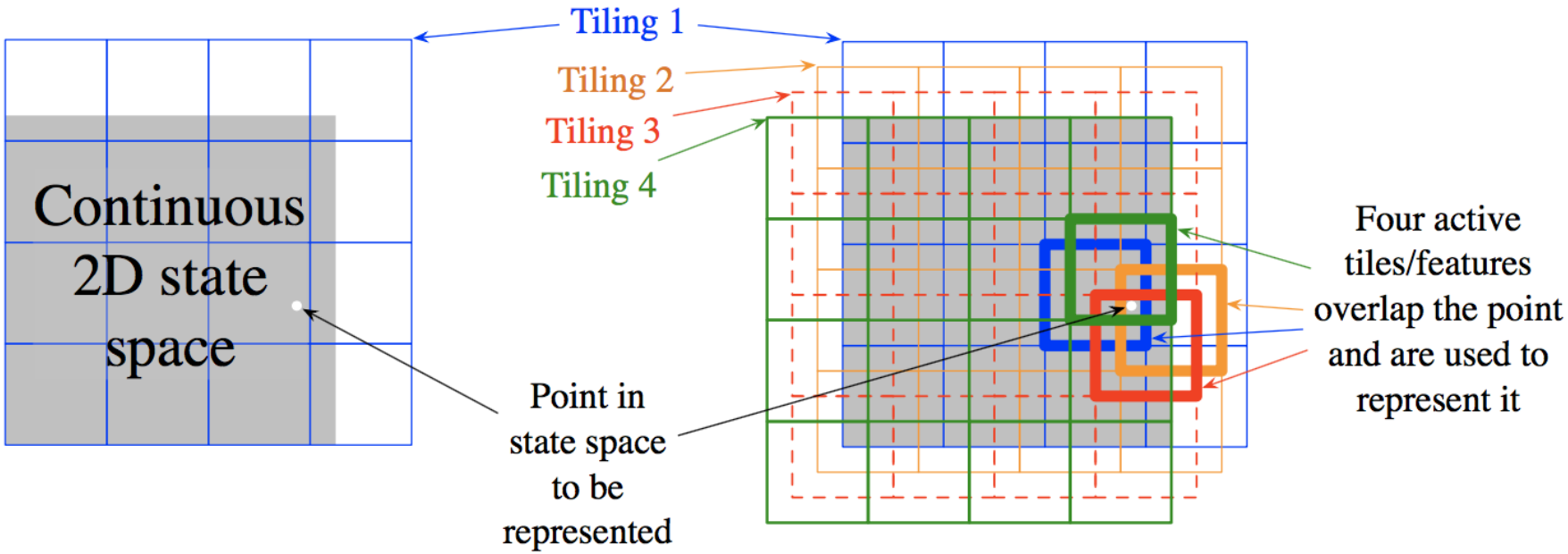
Shallow RL - Tabular Methods

- Tie Breaking
- ~~np.argmax(values)~~

Shallow RL - Linear Function Approximation

- $v(s) = \boldsymbol{w}\phi(s)$
- Polynomials
- Fourier Basis
- Coarse Coding
- Tile Coding
- ...

Shallow RL - Tile Coding



Tile3

`http://incompleteideas.net/sutton/tiles/tiles3.html`

Sutton and Barto (1998)

Deep RL

- Atari Games
- Continuous Action
- Debug and Tuning
- Parallelization

Deep RL - Atari Games

State:

- rescale
- frame skip
- max-pooling across time

Reward:

- clip reward (np.sign ~~np.clip~~)

Value function:

- convolutional network

Actor-Critic:

- shared convolutional network

Mnih et al. (2015)

Deep RL - Continuous Action

High dimensional action $\boldsymbol{a} \in \mathbb{R}^d$
Benchmarks:

- Box2D
- Roboschool
- ~~Mujoco~~

State & Reward:

- normalize by running statistics, i.e $(x - \mu)/\sigma$

Deep RL - Continuous Action

Actor:

- Gaussian distribution $\mathcal{N}(\mu, \mathcal{I}\sigma^2)$
- shared layers for μ and σ
- unbounded μ + unit σ
- bounded μ + unbounded/bounded σ
- ~~unbounded μ + unbounded σ~~

Actor/Critic:

- disjoint fully-connected networks
- even single layer works well

Deep RL - Continuous Action

- low-dimensional state space
- fast enough and challenging enough

Deep RL - Debug and Tuning

NAN:

- decrease learning rate
- divided by zero / log zero
- L2 regularization
- clip something, e.g. gradient, importance sampling ratio, loss

Some hyper-parameters:

- discount ratio: $\star$
- GAE decay ratio: $\star$
- rollout length: $\star\star$
- entropy penalty weight: $\star\star\star$

Deep RL - Parallelization

Limited to multi-process
Possible strategies:

- distributed workers, centralized gradient update
- distributed workers, distributed gradient update
  - rollout-based: on/off-policy method
  - replay-based: off-policy method

Deep RL - Parallelization

- shared statistics v.s. separate statistics
- $\delta = \mu_B - \mu_A$
  $\mu = \mu_A + \delta \frac{n_B}{n_A + n_B}$
  $v = v_A + v_B + \delta^2 \frac{n_A n_B}{n_A + n_B}$

Chan et al. (1982)

Deep RL - Parallelization

With PyTorch:

- set OMP_NUM_THREADS to 1
- incompatible with Roboschool in OS X [2]
- v0.1.12 is much faster than v0.2 in terms of multi-processes

[2] https://github.com/openai/roboschool/issues/86

Deep RL - Parallelization

Distributed Proximal Policy Optimization (Heess et al. 2017)

- distributed machines to multiple processes
- KL penalty to clipped penalty (Schulman et al. 2017)
- multiple mini-batch update to one-pass batch update

Deep RL - Parallelization

Distributed Deep Deterministic Policy Gradient

- DDPG (Lillicrap et al. 2015, Silver et al. 2014)
- $\pi : \mathcal{S} \to \mathcal{A}$ ~~$\mathcal{P}(\mathcal{A})$~~
- parallelization strategy
  - ~~rollout~~
  - ~~non-shared replay buffer~~
  - shared replay buffer

Thanks