Off-policy deep reinforcement learning algorithms (seminar)

Oleksii Hrinchuk

Deep Q-Network (DQN)

1. Initialize Q-Network Q_θ , set initial exploration rate $\varepsilon = 1$. Run epsilon-greedy policy w.r.t. Q_θ (or random policy which is the same) for a number of steps to initialize replay buffer $\mathcal{D}$.

2. (Train loop)
   — Make a step in the environment with policy $\pi = \varepsilon\text{-greedy}(Q_\theta)$
   — Put transition $\mathbf{d}_t = \{\mathbf{s}_t, \mathbf{a}_t, \mathbf{r}_t, \mathbf{s}_{t+1}\}$ into replay buffer $\mathcal{D}$
   — If step % q_update_frequency == 0:
   Sample mini-batch of transitions $\{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_B\} \sim \mathcal{D}$
   Update Q-Network weights $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_\theta$
   — If step % target_update_frequency == 0:
   Update target network weights $\tilde{\theta} \leftarrow \theta$
   — Decay exploration rate $\varepsilon \leftarrow \varepsilon - \Delta\varepsilon$

3. Return greedy policy w.r.t. the learned Q-function Q_θ

From “Human-level Control Through Deep Reinforcement Learning”, Mnih et al., Nature (2015)

Improvements to DQN

Double DQN (DDQN)

- Max operator in TD-targets leads to overoptimistic estimates of Q-function.

$$\begin{aligned} y_t &= \mathbf{r}_t + \gamma \max_{\mathbf{a}' \in \mathcal{A}} Q_{\tilde{\theta}}(\mathbf{s}_{t+1}, \mathbf{a}') \\ &= \mathbf{r}_t + \gamma Q_{\tilde{\theta}}(\mathbf{s}_{t+1}, \arg\max_{\mathbf{a}' \in \mathcal{A}} Q_{\tilde{\theta}}(\mathbf{s}_{t+1}, \mathbf{a}')) \end{aligned}$$

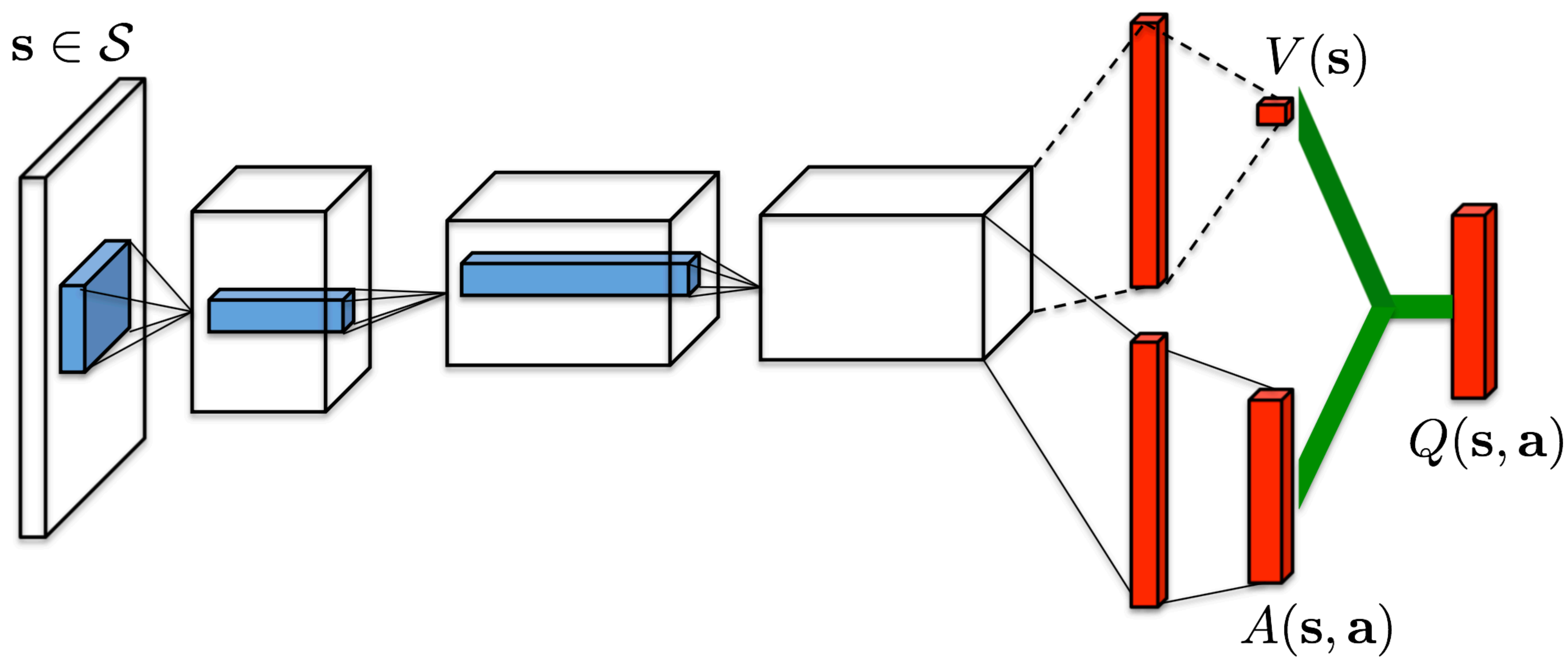
- Q-Network is used to select actions, target network is used to evaluate actions

$$y_t = \mathbf{r}_t + \gamma Q_{\tilde{\theta}}(\mathbf{s}_{t+1}, \arg\max_{\mathbf{a}' \in \mathcal{A}} Q_{\theta}(\mathbf{s}_{t+1}, \mathbf{a}'))$$

From “Deep Reinforcement Learning with Double Q-learning”, van Hasselt et al., AAAI (2016)

Dueling DDQN

- Split Q-function into two channels
  - Action independent value function $V(\mathbf{s})$
  - Action dependent advantage function $A(\mathbf{s}, \mathbf{a})$



$$Q(\mathbf{s}, \mathbf{a}) = V(\mathbf{s}) + A(\mathbf{s}, \mathbf{a}) - \frac{1}{|\mathcal{A}|} \sum_{\mathbf{a} \in \mathcal{A}} A(\mathbf{s}, \mathbf{a})$$

From "Dueling Network Architectures for Deep Reinforcement Learning", Wang et al., ICML (2016)

Prioritized experience replay

- Transitions with high TD-error are sampled more often

$$\mathbf{d}_t = \{\mathbf{s}_t, \mathbf{a}_t, \mathbf{r}_t, \mathbf{s}_{t+1}\}$$

$$\delta_t = \mathbf{r}_t + \gamma \max_{\mathbf{a}' \in \mathcal{A}} Q(\mathbf{s}_{t+1}, \mathbf{a}') - Q(\mathbf{s}_t, \mathbf{a}_t)$$

- Define prioritization
  - proportional $p_t = |\delta_t| + \epsilon$
  - rank-based $p_t = 1/\mathrm{rank}(t)$

- Sample transitions according to probability

$$P(\mathbf{d}_t) = \frac{p_t^\alpha}{\sum_{t'=1}^{|\mathcal{D}|} p_{t'}^\alpha}$$

From “Prioritized Experience Replay”, Schaul et al., ICLR (2016)

Boltzmann exploration

- Epsilon-greedy exploration is bad in situations when picking random action can be disastrous

- Idea: pick exploratory actions based on Q-function:
  — good actions have high probability
  — bad actions have really low probability

$$\pi_{\mathcal{B}}(\mathbf{a}|\mathbf{s}) = \mathrm{softmax}\left(\frac{Q(\mathbf{s},\mathbf{a})}{\tau}\right) = \frac{\exp\left(Q(\mathbf{s},\mathbf{a})/\tau\right)}{\sum_{\mathbf{a}'\in\mathcal{A}} \exp\left(Q(\mathbf{s},\mathbf{a}')/\tau\right)}$$

$$\pi_{\mathcal{B}}(\mathbf{a}|\mathbf{s}) \to \arg\max_{\mathbf{a}\in\mathcal{A}} Q(\mathbf{s},\mathbf{a}), \quad \tau \to 0$$

Additional off-policy references

- A Distributional Perspective on Reinforcement Learning. Bellemare et al., ICML (2017)
- Distributional Reinforcement Learning with Quantile Regression. Dabney et al., AAAI (2018)
- Rainbow: Combining Improvements in Deep Reinforcement Learning. Hessel et al., AAAI (2018)
- Self-Imitation Learning. Oh et al., ICML (2018)

Additional on-policy references

- Asynchronous Methods for Deep Reinforcement Learning. Mnih et al., ICML (2016)

- Trust Region Policy Optimization. Schulman et al., ICML (2015)

- Proximal Policy Optimization Algorithms. Schulman et al. (2017)

Why so many RL algorithms?

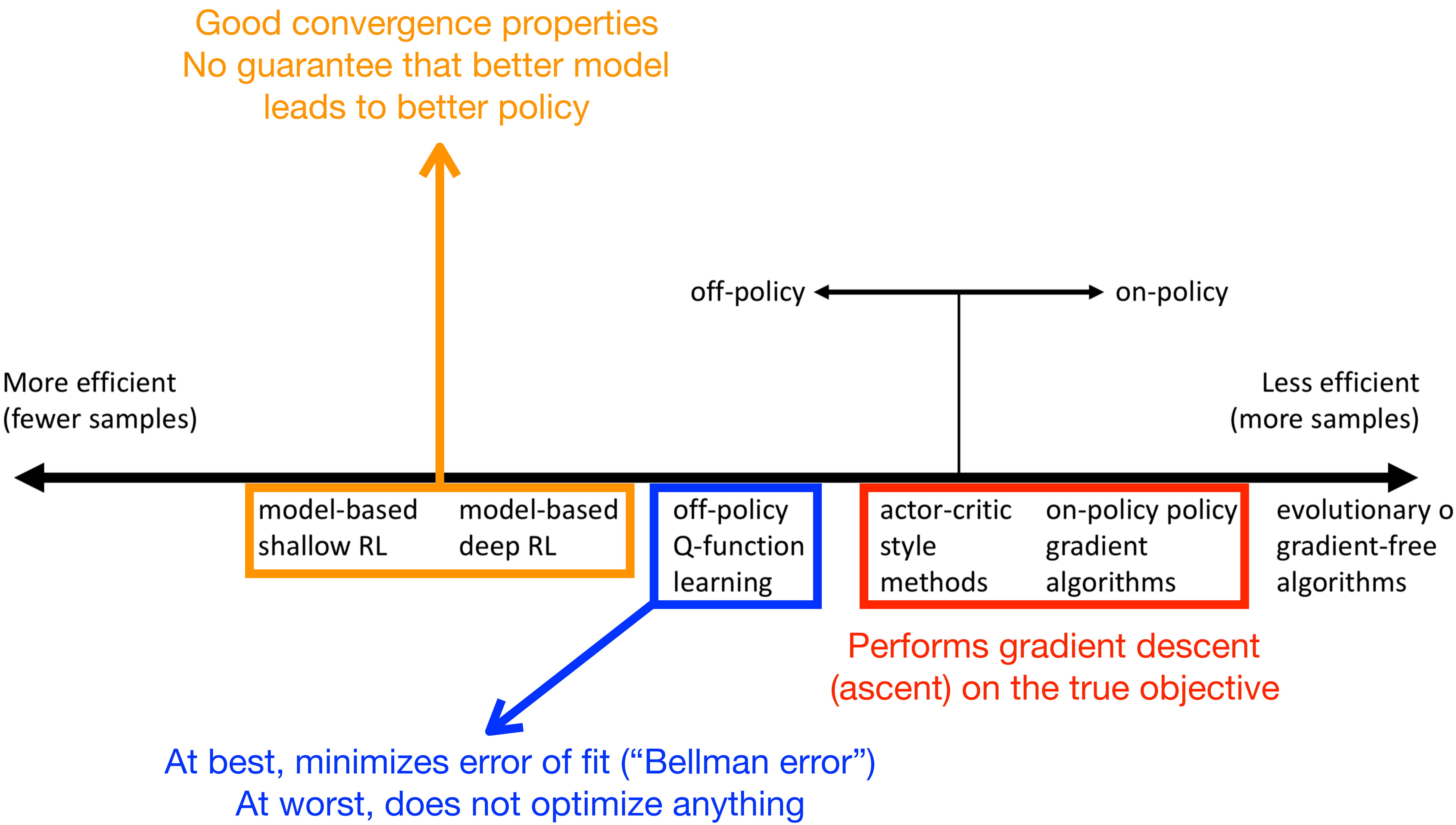
- Different tradeoffs
  — Sample efficiency
  — Stability and ease of use

- Different assumptions
  — Stochastic or deterministic?
  — Continuous or discrete?
  — Episodic or infinite horizon?

- Different things are easy or hard in different settings
  — Easier to represent the policy?
  — Easier to represent the model?

Taken from Sergey Levine UC Berkeley course, Lecture 4: Reinforcement Learning Introduction

Sample efficiency



Taken from Sergey Levine UC Berkeley course, Lecture 4: Reinforcement Learning Introduction

Task

- Train the best agent to play 10x10 Snake game
- You are allowed to adopt third-party open source code, but you have to cite the source
- You are not allowed to change the file snake.py
- You should submit
  — Plot (reward / number of training transitions)
  — Best agent score (mean and std of 1000 runs)
  — Source code which reproduces best agent results
  — Short report on what you try, what worked best (worst)
- Deadline: Sunday, Sep 30, 23:59