



# AIL 722: Reinforcement Learning

## Lecture 38: Continuous Actions, Exploration

Raunak Bhattacharyya



**ScAI**

YARDI SCHOOL OF ARTIFICIAL INTELLIGENCE  
INDIAN INSTITUTE OF TECHNOLOGY DELHI

# Today's Outline

- DQN with continuous actions
- Exploration vs. Exploitation
- Formulating exploration

# Deep Q-Learning with Continuous Actions

$$\pi'(a_t|s_t) = \begin{cases} 1, & \text{if } a_t = \arg \max_{a_t} A^\pi(s_t, a_t) \\ 0, & \text{otherwise} \end{cases}$$

Set  $y_i \leftarrow r(s_i, a_i) + \gamma \cdot \max_{a'_i} Q_\phi(s'_i, a'_i)$

**How do we find the max when we have continuous actions?**

# Sampling-based Approaches

$$\max_a Q(s, a) \simeq \max\{Q(s, a_1), \dots, Q(s, a_N)\}$$

- Derivative free stochastic optimisation

**High dimensions. Samples won't be representative enough**

# Learn Approximate Maximiser

Take some action  $a_i$  and observe  $(s_i, a_i, s'_i, r_i)$  and add it to  $\mathcal{B}$

Sample mini-batch  $\{s_j, a_j, s'_j, r_j\}$

Compute  $y_j = r_j + \gamma Q_{\phi'}(s'_j, a'_j)$

$\phi \leftarrow \phi - \alpha \sum_j \frac{dQ_\phi}{d\phi}(s_j, a_j) (Q_\phi(s_j, a_j) - y_j)$

$\theta \leftarrow \theta + \beta \sum_j \frac{d\mu}{d\theta}(s_j) \frac{dQ_\phi}{da}(s_j, a)$

update  $\phi'$  and  $\theta'$

Can also be viewed as deterministic AC algo

# Trivia

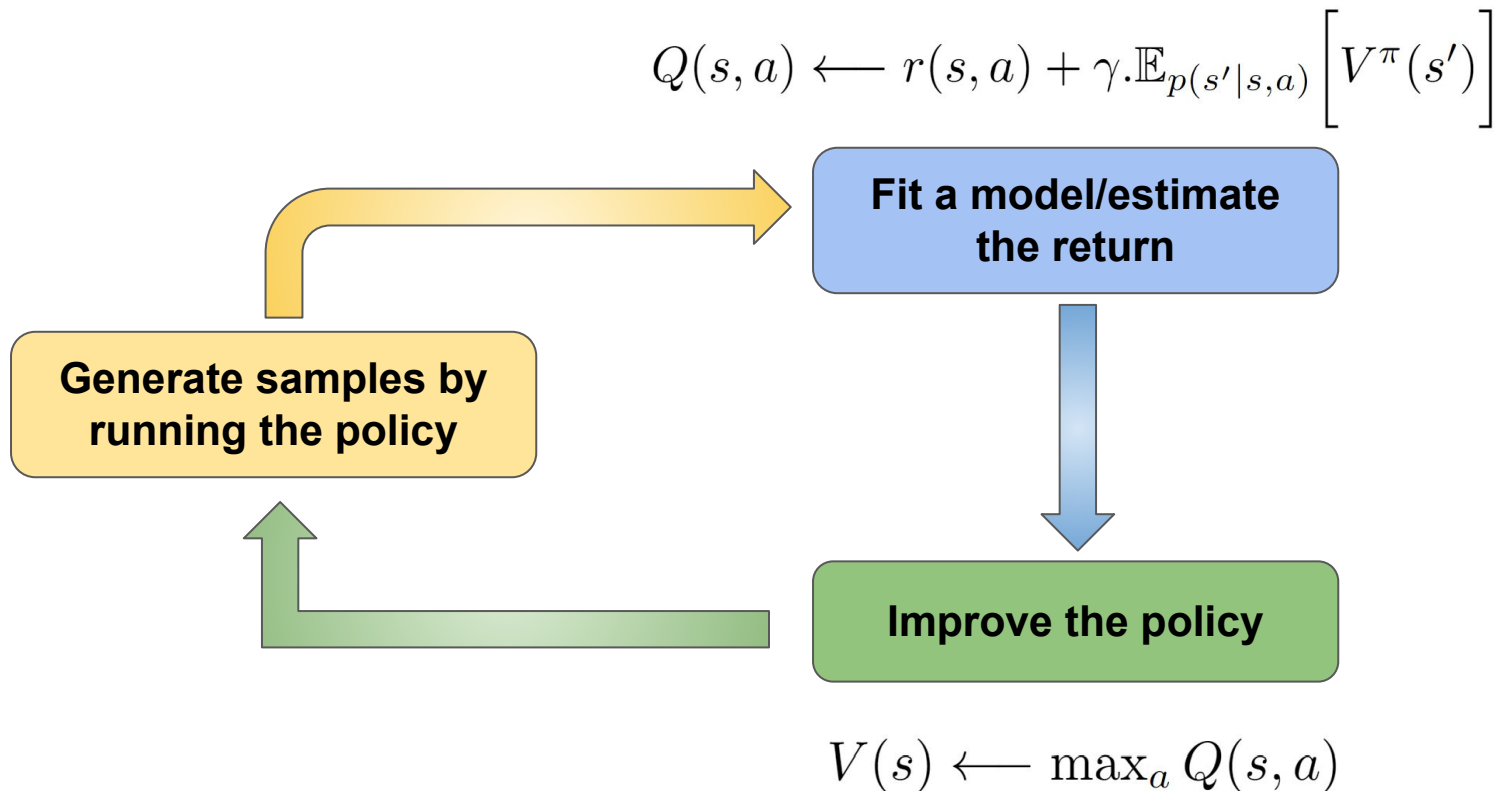
234

R.J. WILLIAMS

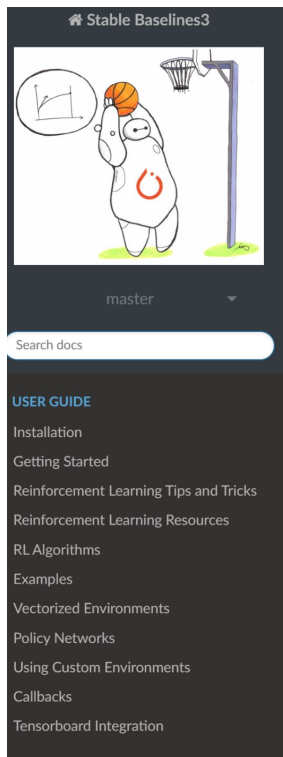
$$\Delta w_{ij} = \alpha_{ij}(r - b_{ij})e_{ij},$$

where  $\alpha_{ij}$  is a *learning rate factor*,  $b_{ij}$  is a *reinforcement baseline*, and  $e_{ij} = \partial \ln g_i / \partial w_{ij}$  is called the *characteristic eligibility* of  $w_{ij}$ . Suppose further that the reinforcement baseline  $b_{ij}$  is conditionally independent of  $y_i$ , given  $\mathbf{W}$  and  $\mathbf{x}^i$ , and the rate factor  $\alpha_{ij}$  is nonnegative and depends at most on  $\mathbf{w}^i$  and  $t$ . (Typically,  $\alpha_{ij}$  will be taken to be a constant.) Any learning algorithm having this particular form will be called a *REINFORCE* algorithm. The name is an acronym for “*REward Increment = Nonnegative Factor  $\times$  Offset Reinforcement  $\times$  Characteristic Eligibility*,” which describes the form of the algorithm.

# Unified View



# Stable Baselines



🏠 / Stable-Baselines3 Docs - Reliable Reinforcement Learning Implementations [View page source](#)

## Stable-Baselines3 Docs - Reliable Reinforcement Learning Implementations

Stable Baselines3 (SB3) is a set of reliable implementations of reinforcement learning algorithms in PyTorch. It is the next major version of [Stable Baselines](#).

Github repository: <https://github.com/DLR-RM/stable-baselines3>

Paper: <https://jmlr.org/papers/volume22/20-1364/20-1364.pdf>

RL Baselines3 Zoo (training framework for SB3): <https://github.com/DLR-RM/rl-baselines3-zoo>

RL Baselines3 Zoo provides a collection of pre-trained agents, scripts for training, evaluating agents, tuning hyperparameters, plotting results and recording videos.

SB3 Contrib (experimental RL code, latest algorithms): <https://github.com/Stable-Baselines-Team/stable-baselines3-contrib>

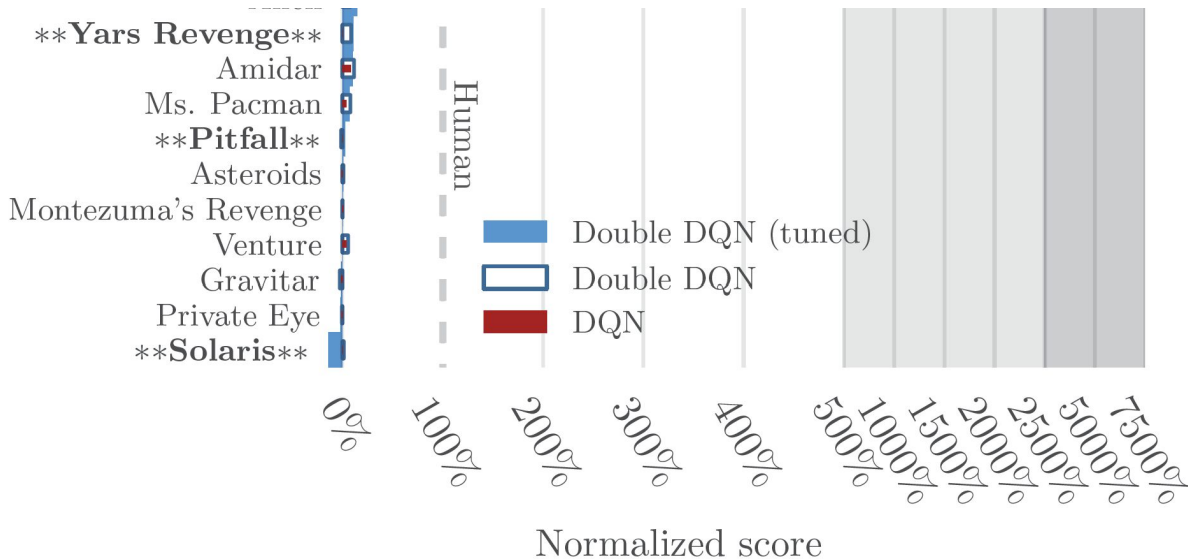
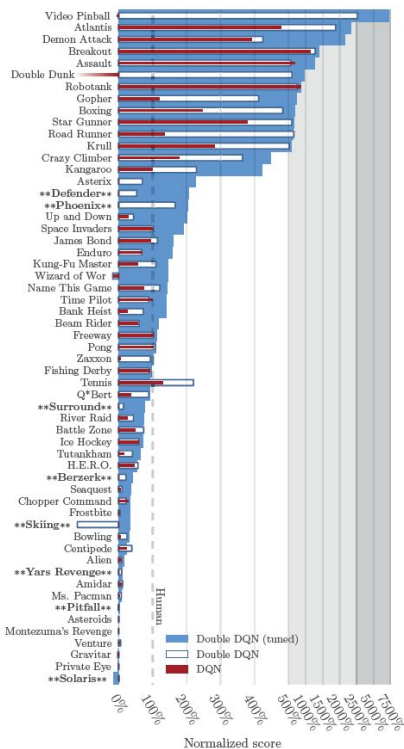
### Main Features

- Unified structure for all algorithms
- PEP8 compliant (unified code style)
- Documented functions and classes
- Tests, high code coverage and type hints
- Clean code
- Tensorboard support
- **The performance of each algorithm was tested** (see *Results* section in their respective page)



# Exploration

# Games where DDQN fails





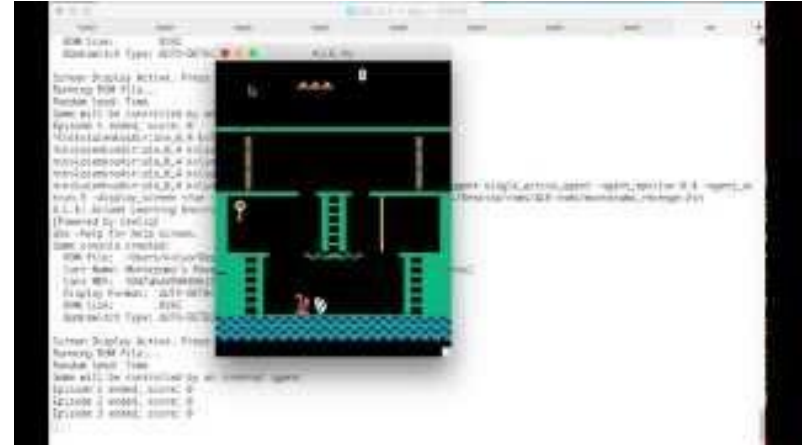
Montezuma's Revenge, Source: [Youtube](#)



Pitfall, Source: [Youtube](#)



Venture, Source: [Youtube](#)



Breakout vs MR, Source: [Youtube](#)

# Exploration vs Exploitation

How can an agent decide whether to attempt new behaviors (to discover ones with higher reward) or continue to do the best thing it knows so far?

**Exploitation: Do what you think will yield the highest reward**

**Exploration: Do the things that you haven't done before hoping that it will yield even higher reward**

Dynamic and persistent decision we have to keep making

**What was the requirement for Q-learning to converge to the optimal state-action value function?**

# Multi-armed Bandits



One-arm bandit, Source: [Youtube](#)



Multi-armed Bandits, Source: [Wikipedia](#)