Reinforcement Learning

Aldo Faisal with contributions
by Ed Johns and Paul Bilokon

Imperial College London

October 2022 – Version 8.3

Revisiting continuous actions I

- In classic value-based Q learning and the DQN case it was easy to find the optimal action, this is non-trivial for general continuous action policies.
- In the case of optimal deterministic policies $\pi^*(s) = a^* = \arg\max_a Q^*(s, a)$.
- When there are a finite number of discrete actions, finding the max is easy, because we can just compute the Q-values for each action separately and pick the largest.
- Conveniently it gives us also the action which maximises the optimal $Q(s, a)$ value.

Revisiting continuous actions II

- In continuous action spaces evaluating the max can become very expensive as we have to search the entire set of possible actions a by some optimisation method. This can become quickly prohibitive when the agent has to do this for every single time step.
- But if we the action space is continuous **and** the function $Q^*(s, a)$ is differentiable, then we can define an efficient, gradient-based learning rule to a deterministic policy $\mu(s) = a$.
- Then we can, instead of running an expensive optimisation each time we wish to compute $\max_a Q(s, a)$, instead just approximate the maximum using $\max_a Q^*(s, a) \approx Q(s, \mu(s))$.
- We will see next in the case of DDPG how this can be done.

DDPG (Lillicrap et al.,2015) I

Deep Deterministic Policy Gradients (DDPG) is a deterministic policy, model-free, off-policy, actor-critic algorithm which

- learns the value model (Q-function) and a policy π.
- learns by off-policy methods and uses the Bellman equation to learn the Q-function (boostrapping part 1)
- uses the Q-function to learn the policy π (bootstrapping part 2)
- is considered the continuous action version of the DQN.

We will next look first at the algorithm (as re-printed from Lillicrap et al, 2015) and then what ideas it contains.

DDPG (Lillicrap et al.,2015) I

Algorithm 1 DDPG algorithm

```
Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$.
Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer $R$
for episode = 1, M do
    Initialize a random process $\mathcal{N}$ for action exploration ①
    Receive initial observation state $s_1$
    for t = 1, T do
        Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise ①
        Execute action $a_t$ and observe reward $r_t$ and observe new state $s_{t+1}$
        Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$ ②
        Sample a random minibatch of $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $R$ ②
        Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
        Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$ ③
        Update the actor policy using the sampled policy gradient:
            $$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \underbrace{\nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}}_{\text{Deterministic version of PG Theorem}}$$ ④
        Update the target networks: ⑤
            $$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
            $$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$
    end for
end for
```

DDPG (Lillicrap et al.,2015) I

To get DDPG to work well a few special elements were introduced:

1. Exploration in continuous action spaces
   - Use Gaussian distributed noise that perturbs a mean action $\mu'(s) = \mu_\theta(s) + \mathcal{N}$ (i.e. Gaussian Policies)
   - This is almost equivalent to discrete action ϵ-greediness (only that we explore more often and much closer to the intended action μ).
2. To control learning variability DDPG uses replay buffers/minibatches & target networks (just as in DQN)

DDPG (Lillicrap et al.,2015) II

3. Error-based learning on the mean-squared Bellman error (MSBE)
    - MSBE (here denoted by L, tells us how closely our approximation of Q comes to satisfying the Bellman equation (compare this to function approximation Q-learning/DQN). $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta_Q))^2$
    - Note, that Lillicrap's notation in their pseudocode differs from our notation, where they denote by θ_Q the parameters of the value function approximation (we used w before) and by θ_μ the parameters of the policy function approximation (we used θ)
4. Policy gradient based learning on the actor network
    - Here we are using a deterministic action version of the PG theorem, which proof (not examinable here) was published in Silver et al (2014).

DDPG (Lillicrap et al.,2015) III

- DQN updates the target network in jumps with many episodes remaining frozen. This does not work well in continuous control cases, where small changes in policy need to remain linked to small changes in control.
- DDPG therefore uses "smooth" updates on the parameter vector θ (that is shared by both actor and critic) which updates slowly but steadily over steps t: $\theta^{t+1} = \beta\theta^t + (1-\beta)\theta'^t, 0 < \beta \ll 1$
- The parameter β controls the degree of blending (or forgetting) between the current parameter estimate θ and its update θ' at t to determine the parameter vector at the next $t+1$.

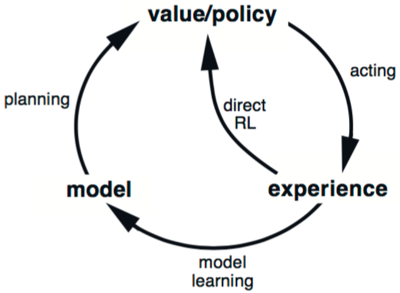
Section overview

Closing thoughts

1 Motivation
2 Reinforcement Learning 101
3 Lets go Markov
4 Markov Decision Process
5 Dynamic Programming
7 Model-Free Control
8 Function Approximation
9 Deep Q Learning
10 Policy Gradients
11 Actor-Critic Methods
12 Tricks of the Trade

Links and Counter-links

- Bootstrapping vs Sampling
- Tabular-Methods vs Function-Approximation Methods
- Model-Free vs Model-Based (outlook)



- Hierarchical RL (outlook)
- Partial Observability (outlook)

The future is in Representation

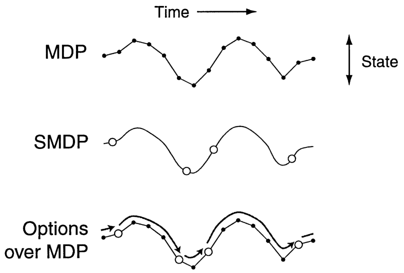
- Better representations of state – the original Deep RL and Partially Observable Markov Decision Processes (POMDPs)
- Better representations of the world and world-models – model-based RL and actor-critics
- Better representations of actions...

Representation for Temporal abstracts: Actions, Options, and Action Grammars I

Consider a traveler deciding to undertake a journey to a distant city.

- To decide whether or not to go, the benefits of the trip must be weighed against costs.
- choices must be made at each leg of the trip, e.g., whether to fly or to drive, whether to take a taxi or to arrange a ride
- choices what bus to take, what taxi company to hire

Representation for Temporal abstracts: Actions, Options, and Action Grammars II



Where to learn from here

If you want to learn more look out for the latest work on Arxiv.org (this is not peer-reviewed, so bogus or poor work may be shown here, but it is always the most up to date). Much better for quality are the following top conferences to follow (publications in their annual proceedings or their presentations, sometimes online):

- NeuriIPS (biggest RL meeting 1000 attendees in the RL workshop)
- ICML
- ICLR (deep learning)
- RLDM (also more neuroscience/psychology)
- IROS (top robotics conference)

THEY GIVE YOU A LOT OF TREATS WHILE THEY'RE TRAINING YOU, SO PLAY DUMB FOR AS LONG AS YOU CAN.
guy & rodd 11-24
facebook.com/brevitycomic