# Notes on Batch Reinforcement Learning

(a chapter from the Book *“Reinforcement Learning: State of the art”,* compiled by Marco Wiering and Martijn van Otterlo, 2012)

*Notes taken by D. Gueorguiev, Jan 26, 2024*

## Motivation for taking the Notes

The book [“*Reinforcement Learning: State of the art*”](https://github.com/dimitarpg13/reinforcement_learning_and_game_theory/blob/main/books/Reinforcement_learning_state_of_the_art.pdf) appeared in 2012 at the dawn of the Reinforcement Learning revolution. Note that in that year articles such [“Playing Atari with Deep Reinforcement Learning”](https://github.com/dimitarpg13/reinforcement_learning_and_game_theory/blob/main/articles/ReinforcementLearning/Playing_Atari_with_Deep_Reinforcement_Learning_Mnih_2013.pdf) by Volodymyr Mnih et al, [“Human-Level Control through Deep Reinforcement Learning”](https://github.com/dimitarpg13/reinforcement_learning_and_game_theory/blob/main/articles/ReinforcementLearning/DQNNaturePaper_Mnih_2015.pdf) by Volodymyr Mnih et al, [“Continuous Control with Deep Reinforcement Learning”](https://github.com/dimitarpg13/reinforcement_learning_and_game_theory/blob/main/articles/ReinforcementLearning/Continuous_control_with_deep_reinforcement_learning_Lillycrap_2015.pdf) by T. P. Lillicrap et al, [“Trust Region Policy Optimization”](https://github.com/dimitarpg13/reinforcement_learning_and_game_theory/blob/main/articles/ReinforcementLearning/TrustRegionPolicyOptimization_Schulman_2015.pdf) by John Schulman et al, [“Deep Reinforcement Learning with Double Q-learning”](https://github.com/dimitarpg13/reinforcement_learning_and_game_theory/blob/main/articles/ReinforcementLearning/Deep_Reinforcement_Learning_with_Double_Q-learning_Hasselt_2015.pdf) by Hado van Hasselt et al, [“Proximal Policy Optimization Algorithms”](https://github.com/dimitarpg13/reinforcement_learning_and_game_theory/blob/main/articles/ReinforcementLearning/Proximal_Policy_Optimization_Algorithms_Shulman_2017.pdf) by John Schulman et al had not been published yet. Clearly, there has been a significant (r)evolution in terms of terminology, notation, and theory relevant to Offline Reinforcement Learning in the last 10 years and the intent of this Notes is to capture the momentum of this (r)evolution as well as to look closely into the state-of-the-art Offline Reinforcement Learning theory from 10 years ago.

Note: *Batch Reinforcement Learning* is an older term for *Offline Reinforcement Learning*.

## Introductory Notes

Historically, the term *Batch* Reinforcement Learning is used to describe RL setting, where the complete amount of learning – usually a set of transitions sampled from the system – is a priori given and fixed. This semantics is aligned with the standard definition of the algorithmic term *Batch* (or *Offline*) *mode of execution*.

The task of the learning system then is to derive a solution – usually an optimal policy – out of this given batch of samples. In the [namesake chapter by Sascha Lange et al](https://github.com/dimitarpg13/self_supervised_learning/blob/main/literature/Lange_Gabel_EtAl_RL-Book-12.pdf), the assumption of an a priori fixed training set is relaxed.

The batch RL algorithms are characterized by two basic ingredients: all observed transitions are stored, and updates occur synchronously in batch (or offline). This allows the definition of batch methods, that are allowed to grow the set of sample experience, in order to incrementally improve their solution. From the interaction perspective, the incremental batch approach reduces the difference between batch methods and pure online learning methods. The Batch RL algorithms attract growing interest due to the fact that basic algorithms like Q-learning usually need many interactions until convergence to good policy is achieved. Ideas from the batch RL usually converge faster than standard Q-learning algorithms.

## Appendix

### Notation and Definitions

**Distribution of the Dynamics of the MDP**: defined through the following 4 arguments function:

which is the probability to get from state to state with action and with reward .

**Distribution of the state-transition probabilities**: defined through the following 3 arguments function:

**Markov Decision Process** (abbrev *MDP*): a 5-tuple with

* is a set of states (finite or infinite, discrete, or continuous)
* is a set of actions (finite or infinite, discrete, or continuous)
* is the function which describes the MDP dynamics i.e. probability to get from state to state with action and with reward .
* defines a *reward function*
* is the discount factor which determines to what extent the focus is on the most recent rewards. with there is no focus on the most recent rewards only.

Note: There is another equivalent definition of Markov process which uses the *state-transition probabilities distribution* represented by the three-argument function . With this definition the Markov Decision Process is defined as a 5-tuple , where:

* is a set of states (finite or infinite, discrete, or continuous)
* is a set of actions (finite or infinite, discrete, or continuous)
* is the distribution of the *state-transition probabilities* i.e. the probability to get from state to state with action .
* defines a *reward function*
* is the discount factor

Note 2: A more detailed definition of MDP involves specifying the initial state distribution and augments either of the MDP definitions as:

Markov Decision Process with specified *initial state* is a 6-tuple , where:

* is a set of states (finite or infinite, discrete, or continuous)
* is a set of actions (finite or infinite, discrete, or continuous)
* is the probability to get from state to state with action .
* defines a *reward function*
* defines the initial state distribution
* is the discount factor

**Learning Policy** (or just *Policy*): function which represents mapping from states to probabilities of selecting each possible action.

If the agent is following policy at time , then is the probability that if . Note that is an ordinary function which defines a probability distribution over for each .

We would like to modify the policy with training or experience.

### State-Value and State-Action Functions

Let us assume that the current state is , and actions are selected according to a stochastic policy . Then we would like to derive an expression for the expectation of in terms of and .

Recall, the function defines the dynamics of the MDP and is given as:

for all (1)

Then we can write:

(2)

Here denotes the reward of going from state to state taking action is given by MDP’s function: .

**State-Value Function for Policy** (or simply *Value function;* aka *function*): the value function of a state under a policy , denoted with , is the expected return when starting in and following thereafter. For MDPs, we can define formally by

for all (3)

where denotes the expected value of a random variable given that the agent follows policy , and is any time step. Note that the value of the terminal state, if any, is always zero.

**Action-Value Function for Policy** (aka *function*):

We define the value of taking action in state under a policy , denoted , as expected return starting from , taking the action , and thereafter following policy :

for all and (4)

Let us express in terms of and . Given a state s, the state value function , given with (3), is equal to the expected cumulative return from that state given a distribution of actions . The action value function is the expectation of the return given state , and taking action as a starting point, and following policy thereafter. Therefore, given a state the action-value function is the weighted sum of the action-values over all relevant actions weighted by the policy weight:

(5)

Given a state and an action let us express the action-value function in terms of the state value function and the function defining the MDP dynamics . Recall, given a state and an action , the action value function is given by the mathematical expectation of the discounted future rewards i.e. return . The return is the discounted sequence of rewards after the time step and it can be written as:

(6)

It is important to recognize that

. (7)

The first term on the right-hand side of (7) can be expressed as:

. (8)

As before, denotes the reward of going from state to state taking action is given by MDP’s function: .

The expectation in the second term on the right-hand side of (8) can be expressed as:

. (9)

This is the expectation of the return starting at the next time step following the policy given the current state and the action , chosen according to .

Substituting (8) and (9) into (7) gives us:

. (10)

Thus, the action-value function given state s and action following policy is expressed as the sum of the next reward and discounted state-value weighted by probability distribution over the possible next states and next rewards from the given action and state .

## Bibliography

[Batch Reinforcement Learning, Sascha Lange, Thomas Gabel, Martin Riedmiller](https://github.com/dimitarpg13/self_supervised_learning/blob/main/literature/Lange_Gabel_EtAl_RL-Book-12.pdf), a chapter from the book “Reinforcement Learning: State-of-the-Art”, 2012