# Note on functions and functions in Reinforcement Learning

(from Chapter 3 of *Sutton and Barto’s* book)

Table of Contents

[Note on functions and functions in Reinforcement Learning 1](#_Toc151875271)

[Notation and Definitions 1](#_Toc151875272)

[State-Value and State-Action Functions 1](#_Toc151875273)

[Estimation Methods for State-Value and State-Action Functions 2](#_Toc151875274)

[Bellman’s Equations for State-Value and State-Action Functions 3](#_Toc151875275)

[Bellman’s equation for action values 3](#_Toc151875276)

## Notation and Definitions

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

* is a set of states (finite or infinite)
* is a set of actions (finite or infinite)
* is the probability to get from state to state with action .
* is the immediate reward after going from state to state with action .
* 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.

**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)

**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 s 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** (or simply *Value function;* 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 a 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)

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 action-value weighted by probability distribution over the possible next states and next rewards from that action and state .

### Estimation Methods for State-Value and State-Action Functions

The value functions and can be estimated from experience. For example, if an agent follows policy and maintains an average, for each state encountered, of the actual returns that have followed that state, then the average will converge to the state value , as the number of times that state is encountered approaches infinity. If separate averages are kept for each action taken in each state, then these averages will similarly converge to the action values, . We call estimation methods of this kind *Monte Carlo methods* because they involve averaging over many random samples of actual returns. Of course, in case there is a large number of states then it would not be feasible to manage separate averages for each state. Instead the agent would have to maintain and as parametrized functions (with fewer parameters than states) and adjust the parameters to better match the observed returns. This approach can produce accurate estimates, although much depends on the nature of the parametrized function approximator.

## Bellman’s Equations for State-Value and State-Action Functions

The value functions satisfy recursive relationships this property of value functions will prove quite useful.

For any policy and for any state s , the following consistency condition holds between the value of s and the value of its successor states. Starting with (6) applied to the definition of :

(11)

Using (5) the last equation becomes:

for all (12)

where it is implicit that the actions, , are taken from the set , that the next states, , are taken from the set , and that the rewards, , are taken from the set . Note that the right-hand side of (12) is interpreted as an expected value obtained as a sum over the values of the triplet , , and . For each triplet the quantity is weighed by its probability, .

Eq. (12) is known as the *Bellman equation* for . It expresses a relationship between the value of a state and the values of its successor states.

Figure 1: Backup diagram for

This relationship is expressed by the *Backup diagram* shown on Figure 1. Each open circle, which will be denoted as *state node* so forth, colored in blue represents a state (shown with ). Each solid circle, colored in blue represents a state-action pair and will be denoted as *state-action node* so forth. The specific state on the rightmost state-action node is shown as . Each directed blue edge connects state node with state-action node and represents application of the policy to the state node . Each solid circle colored in red represents *reward-state node*. Each directed red edge emanating from a state-action node ends in a possible reward-state node corresponding to specific probable pair of reward and new state . Thus, each directed red edge represents the application of the function of the MDP dynamics. The Bellman equation (12) averages over all of the possibilities weighing each possibility represented by a path from the root of the Backup diagram on Figure 1 to a leaf by its probability of occurring. It states that the value of the start state must equal the discounted value of the expected next state plus the reward expected along the way. The value function is the unique solution to its Bellman equation. Various methods exist to compute exactly, approximate, or learn the value function.

### Bellman’s equation for action values

Let us derive a similar recursive relation with respect to the state-action value function. That is, we will find out what is the relation between the action value and that for the possible successors to the state-action pair - . The derivation follows from the Backup diagram shown on Figure 2 below.

Figure 2: Backup diagram for

From (7) we can write:

(13)

The expectation of the reward on the right-hand side can be rewritten as:

(14)

Here using (7) again we denote with the expression for .

Thus we get the Bellman’s equation with respect :

(15)