Continuous State Spaces

Classic Control Tasks

CartPole

A pole is mounted on the cart, and you are supposed to apply force to the cart (in either direction) to balance the pole and keep it standing. The longer you can balance the pole, the more reward you get.

Cart Pole State Space

State Component	Min	Max
Cart Position	-4.8	4.8
Cart Velocity	-Inf	Inf
Pole Angle	-0.418 rad (-24 deg)	0.418 rad (24 deg)
Pole Angular Velocity	-Inf	Inf

Cart Pole Action Space

Value	Action
0	Push cart to the left
1	Push cart to the right

Acrobot

Agent has to swing the double pendulum to touch the bar above with its tip. The faster (lesser time) the tip touch the bar, the more reward you get; in other words, the longer it takes, the more penalty you get.

Acrobot State Space

State Component	Min	Max
Sine of first joint	-1	1
Cosine of first joint	-1	1
Sine of second joint	-1	1
Cosine of second joint	-1	1
Angular Velocity of first joint	-Inf	Inf
Angular Velocity of second joint	-Inf	Inf

Acrobot Action Space

Value	Action
0	Apply +1 on the torque between the links
1	No force applied

Value	Action
2	Apply -1 on the torque between the links

Mountain Car

Agent has to swing the car trapped in a valley to touch the flag on top of the slope on the right side. The faster (lesser time) the car reaches the flag, the more reward you get; in other words, the longer it takes, the more penalty you get.

Mountain Car State Space

State Component	Min	Max
Car position (on horizontal axis)	-1.2	0.6
Car velocity	-0.07	0.07

Mountain Car Action Space

Value	Action
0	Accelerate to the left
1	No acceleration
2	Accelerate to the right

Pendulum

Agent has to keep the pendulum upright by applying force. The longer you can balance the pendulum upright, the more reward you get.

Pendulum State Space

State Component	Min	Max
Sine of angle of the pendulum	-1	1
Cosine of pendulum	-1	1
Angular Velocity of pendulum	-Inf	Inf

Pendulum Action Space

Value	Action
[-2,2]	Torque applied on the pendulum

State Aggregation

Tabular methods are only good for Discrete State and Action spaces, because there are infinite possible states in Continuous State spaces or infinite possible actions in Continuous Action spaces. As such, we need to transform/discretize the continuous state into a discrete representation, or use other algorithms capable of dealing with continuous state and action spaces.

Transforming states

Convert continuous state space into discrete state space:

$$\mathrm{R} \rightarrow \{S_0, S_1, S_2, ..., S_N\}. \tag{1}$$

State Aggregation

Grouping values within a sub-range into a single state. This will limit the precision of our estimate because some different values will now share the same state. On the other hand, if you increase the number of state for improved precision, it will be harder to learn because you will end up with a larger number of states.

$$[0, 20] \rightarrow \begin{cases} S_0 & \text{for } [0, 2), \\ S_1 & \text{for } (2, 4), \\ S_2 & \text{for } (4, 6), \\ ... & \\ S_9 & \text{for } (18, 20] \end{cases} \tag{2}$$

This also works where the state has several dimensions, where you just aggregate each of the dimension and get the Cartesian product of the sets of aggegated dimensions.

Tile Coding

With state aggregation, there is a loss of precision due to Discretization Error. So, we perform several state independent aggregations where each one aggregate a different range of values. We then take the average of the state values from all aggregations.