Reinforcement Learning: Fundamentals

Earl Wong

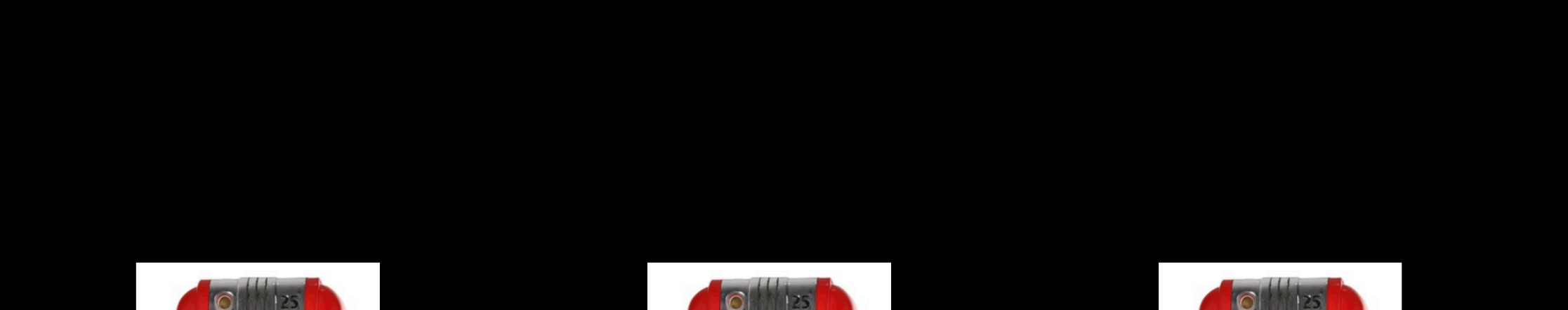
Multi-Arm Bandit

25 cents per play

- Consider the following gambling problem.
- You have three slot machines in front of you.
- Which one should you play, to maximize your monetary return?
- How would you answer this question?

Multi-Arm Bandit

- You play machine one four times and receive the following payout: 0, 0.25, 0, 0.25
- You play machine two four times and receive the following payout: 0.75, 1, 1, 1
- What should you do next? Should you only play machine two from this point on? Or, should you play machine three four times?
- You decide to play machine three four times and receive the following payout: 0, 0.25, 0.25, 10.00



Statistical distribution of the payouts after 4 plays on each machine.

Machine three has the highest average payout.

Should you play only machine three, from this point on?

Multi-Arm Bandit

- From a statistical standpoint, you need to be confident that machine three will continue to yield the largest return, before “fully committing”.
- i.e. You need to collect more observations / samples from all of the machines to reach this conclusion.
- From a reinforcement learning standpoint, you need to continue EXPLORING all of the actions.

In Words

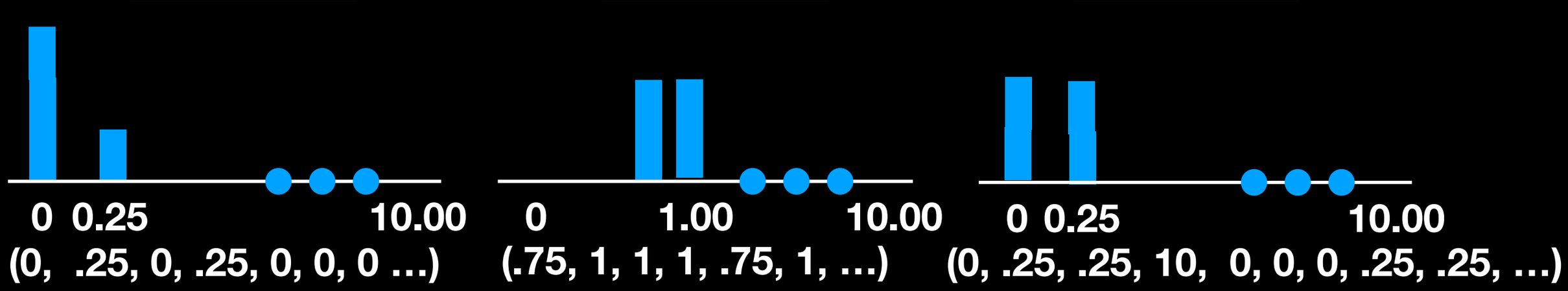
- You (the agent) are making a decision about which slot machine to play. i.e. What action to take.

- You (the agent) are interacting with the environment, which consists of three slot machines.

- The environment is providing you (the agent) with a reward for your action. (The reward could be 0, 0.25, … 10.00)

- Ideally, you want to keep playing every machine (exploring) until you can confidently ascertain which machine produces the largest average payout (expected return).

- Once this is known with high confidence, you can then commit to playing only that machine (exploitation).



Statistical distribution of the payouts after 100,000 plays on each machine.

The expected return for machine 1 is 0.0625.
The expected return for machine 2 is 0.75.
The expected return for machine 3 is 0.125.

With high confidence, you can now begin the exploitation process, by playing only machine two.

Terminology

- Agent: The learner that executes the actions
- Action: The decision that is made
- Reward: A quantity describing an immediate amount that is gained or lost
- Expected Return: The long term average of what is gained or lost
- Exploration: The process of continuing to take all actions / play every machine, to determine the machine with the highest payout.
- Exploitation: The process of playing only the machine with the highest expected return, in order to maximize the payout.

The Concept of State

- In the multi-arm bandit problem, an action was selected and a reward was received.

- In more advanced problems, the concept of “state” enters into the picture.

- What is state?

- Informally, state is defined as “the true description / representation of the environment at a specific time = the complete information available to the agent, about the environment”.

Examples of State

- State could be represented by a low dimensional set:
  S = {Red, Yellow, Green}

- State could be represented by a medium dimensional set:
  S = {0, 1, 2, … 180}, where each number represents the possible angle of a pole.

- State could be represented by a high dimensional set:
  S = {C1, C2, … C(2^64)}, where each Ci represents a specific checker board configuration.

State vs Observation

- At the same time, the representation of the environment may not be complete.
- When this is true, the agent has limited information.
- When this is true, the term observation is used.

State vs Observation

- For example, consider GridWorld.
- State: The entire grid is observed, along with the agent's position.
- Observation: (Only) a 3x3 neighborhood of the agent's position is observed.

State vs Observation

- For example, consider autonomous driving.
- State: The position, speed and trajectory of every car and pedestrian in the scene are known.
- Observation: Not all of the above is known, due to occlusions in the scene.

State vs Observation

- Shortly, we will introduce the Markov Decision Process (MDP).
- MDP's assume full knowledge of state.
- When this is not true, the MDP (then) becomes a partially observed MDP (POMDP), with states being replaced by observations.
- In practice, the RL literature may use a state, action, reward framework, even for problems that involve an observation, action, reward framework.
- This abuse of notation is "understood".

The Markov Property

The Markov property says that the future is independent of the past, given the present.

Now, let's look at the application of the Markov property, in relation to state.

Let S_{t+1}, S_t, S_{t-1} ... represent states at time t+1, t, t-1, …

A state S_t is Markov iff: $\Pr[S_{t+1} \mid S_t] = \Pr[S_{t+1} \mid S_t, S_{t-1}, S_{t-2}, \cdots]$

Under the Markov property, once S_t is known, all previous information is irrelevant.

The Notion of Transition Probability

Given the concept of state and the Markov property, a transition matrix can be defined.

The transition matrix defines the probability of transitioning to a future state s', given the current state s:

$$P = \Pr\left[S_{t+1} = s' \,\middle|\, S_t = s\right]$$

For S = {Red, Yellow, Green}, the transition matrix might look like the following:

	Red	Yellow	Green
Red	0.3	0.6	0.1
Yellow	0.5	0.4	0.1
Green	0.2	0.0	0.8

Return and Reward

Finally, reinforcement learning is predicated on the concept of reward.

In general, you want to take actions that either maximize your return (sum of rewards), or minimize your total loss.

Below, is the equation describing the total return at time t, computed using future, discounted rewards:

Return: $$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \cdots$$

In some situations, the rewards may be sparse (= reward of 0 for many time steps).

Sparse rewards make the reinforcement learning problem significantly more challenging.

Return and Reward

In the previous equation, γ represented a discount factor that takes values in [0, 1].

Discount factors with $\gamma < 1$ are useful because:

1) γ allows you to model / discount future uncertainty.

2) For infinite processes, the return will be (numerically) well behaved.

Finally, we define the Expected Return for a given state s : $E[G_t \mid S_t = s]$

Markov Reward Process

Given these parameters, we can now define a Markov Reward Process as :

$$(S, R(s), P(s' \mid s), \gamma)$$

where S contains a countably, infinite set of states.

Example:

The output of the Markov Reward Process contains a collection of states: Red, Yellow, Green.

… where every state has an associated reward: Red = 5, Yellow = 2, Green = 6 with associated discount factor …

… and known transition probabilities.

Value Function

We now introduce a widely used construct in reinforcement learning - the value function.

$$V(s) = E\left[G_t \,\middle|\, S_t = s\right]$$

$$= E\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots \mid S_t = s\right]$$

$$= E\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right]$$

where E denotes the expectation operator.

The value function provides the following information: What is the value of being in a particular state?

If we are trying to maximize (minimize) expected return, it is important to know the expected return for each state.

Value Function

- In the previous equation, each R denotes the reward obtained at a specific time step, t.

- If the process is stochastic, we repeat the process for every state -averaging the results, to obtain the expected return for that state.

A Different View

Next, let's take a complimentary view of the value function:

$$V(s) = R + \gamma \cdot \sum_{s' \in S} P \cdot V(s')$$

The first term R denotes the expected reward at state s.

The second term computes the weighted average of the expected returns, when transitioning from state s to s'.

The original view -using rollouts and averaging- is a model free view.

The above view -requiring knowledge of parameters P and R- is a model based view.

Markov Decision Process

Next, let's define a set of actions A, and add it to the Markov Reward Process. This defines the Markov Decision Process:

$$(S, A(s), R(s, a, s'), P(s' \mid s, a), \gamma)$$

Redefining R and P, so that R and P also depend on the action, we obtain:

$$R = E[G_t \mid S_t = s, A_t = a]$$

$$P = \Pr\left[S_{t+1} = s' \;\middle|\; S_t = s, A_t = a\right]$$

Policy

In order to maximize the expected return for every state, we need to choose our actions wisely.

i.e. The policy is defined as the rule(s) used by the agent to decide what action to take.

Let a policy π be defined as a distribution of actions, given states.

$$\pi(a|s) = P[A_t = a | S_t = s]$$

The policy fully defines the behavior of the agent:

$$A_t \sim \pi(\cdot | S_t), \forall t > 0$$

Ideally, we would like our agent to select the best possible action for a given state, in order to maximize the expected return.

i.e. The optimal policy.

Value Function

- Now, let's look at value functions, incorporating the notion policy, into the process.

- The state value function is the expected return starting from state s, following the policy π :

$$V_\pi(s) = E_\pi\left[G_t \,\middle|\, S_t = s\right]$$

- The action value function is the expected return starting from state s and action a, acquiring a reward, and then following the policy π :

$$Q_\pi(s,a) = E_\pi[G_t \,|\, S_t = s, A_t = a]$$

Bellman Equations

$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} P(s' \mid s, a)\left[R(s, a, s') + \gamma V_\pi(s')\right]$$

$$Q_\pi(s, a) = \sum_{s'} P(s' \mid s, a)\left[R(s, a, s') + \gamma \sum_{a'} \pi(a' \mid s') Q_\pi(s', a')\right]$$

The Bellman equations (given above) describe the state value function and action value function for a Markov Decision Process.

The Bellman equations are non-linear and have no closed form solution.

As a result, the Bellman equations need to be solved iteratively.

Iterative Solutions

- Iterative solutions such as Value Iteration and Policy Iteration are used to solve the Bellman Equations.

- Value iteration iterates the state value function. Upon convergence, an optimal policy can be determined implicitly.

- Poly iteration switches between policy evaluation (see value iteration) and policy improvement. Upon convergence, the optimal policy is the final policy improvement policy.

- These iterative approaches require knowledge of P and R.

- In practice, we may or may not know P and R.

Optimality

- Recall: We would like to find a policy that maximizes (minimizes) the state value function or action value function.

- If we know the action value functions for different policies:

$$Q_*(s,a) = max_\pi Q_\pi(s,a)$$

- Or, if we know the state value functions for different policies:

$$V_*(s) = max_\pi V_\pi(s)$$

- i.e. We are choosing the policy that yields the maximum (minimum) expected return.

Optimality

- If we know the optimal action value function, we can also determine the optimal policy by performing:

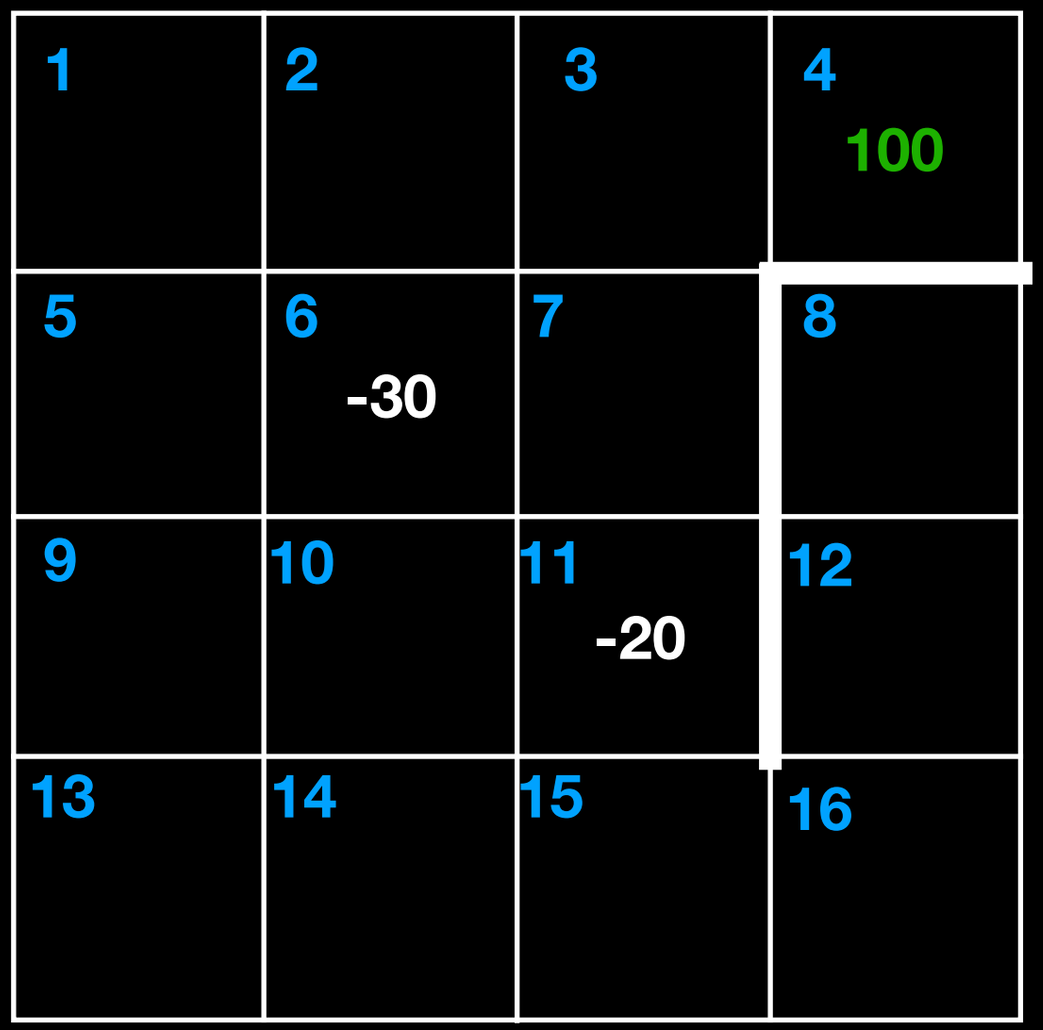
$$\pi_*(s) = \arg\max_a Q_*(s, a)$$

- Similarly, if we know the optimal state value function, we can determine the optimal policy by performing:

$$\pi_*(s) = \arg\max_a \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma V_*(s')]$$

- Note: For any deterministic MDP, there is always an optimal policy.

Grid World - Intuition for State Value Function V(s) Using Rollouts



Problem Statement

Start at any position on the grid (= any state). As shown, there are 16 possible states.

The goal is get to the terminal state (= 4) of the grid, while incurring minimal loss.

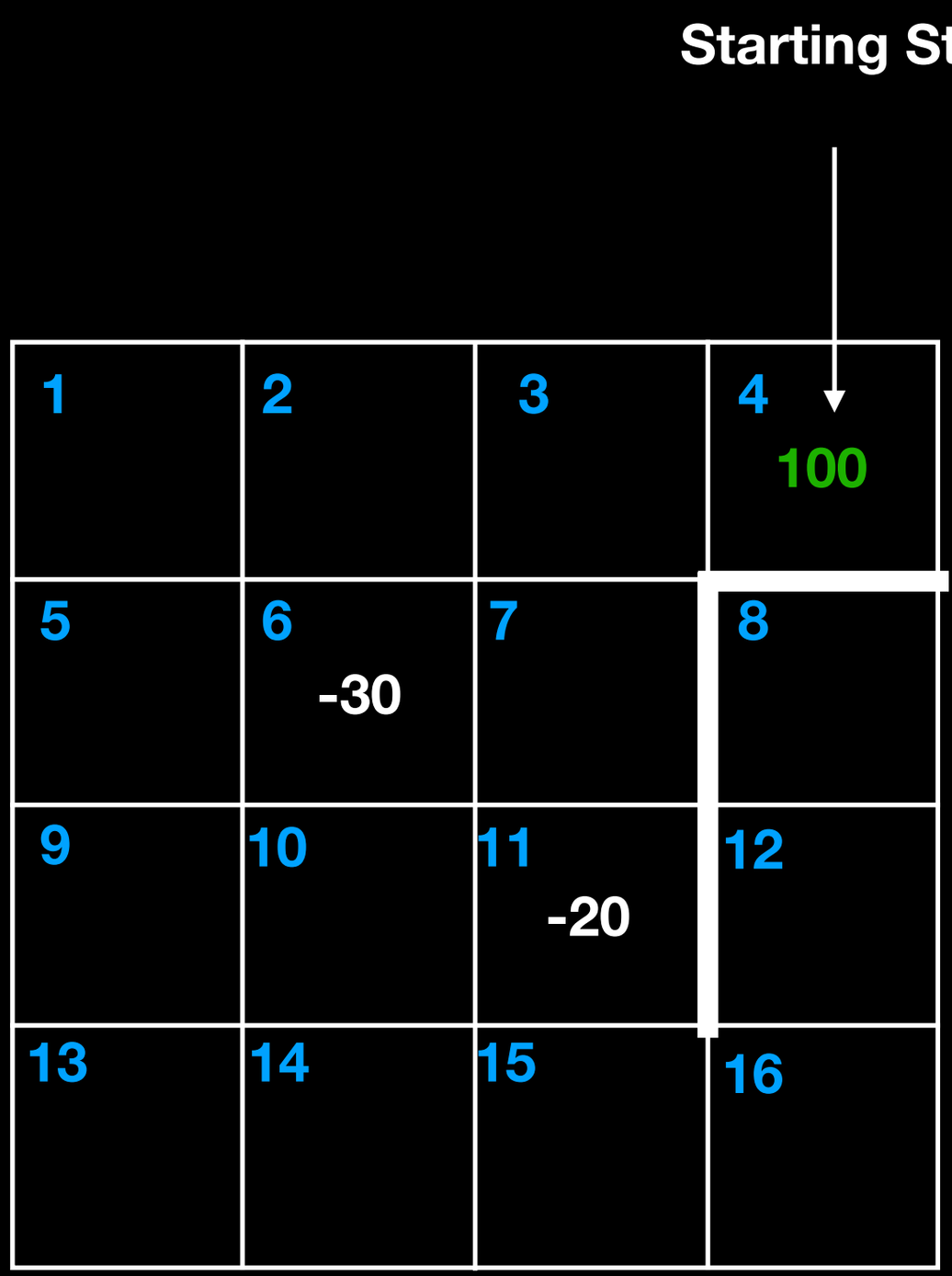
There, the process will end / you will collect an immediate reward of 100.

For every transition from one neighboring state to the next, resulting from an action a, you will collect a reward of -1 (= a loss).

If you move into grid 6 or 11, you will also collect an immediate reward of -20 or -30 (= losses).

You cannot move off the grid or through the solid wall.

Next, we show rollouts, using a random policy.



Rollout 1

Starting State = 4

Immediate Reward = 100

Return = 100

Rollout 2

Starting State = 4

Immediate Reward = 100

Return = 100

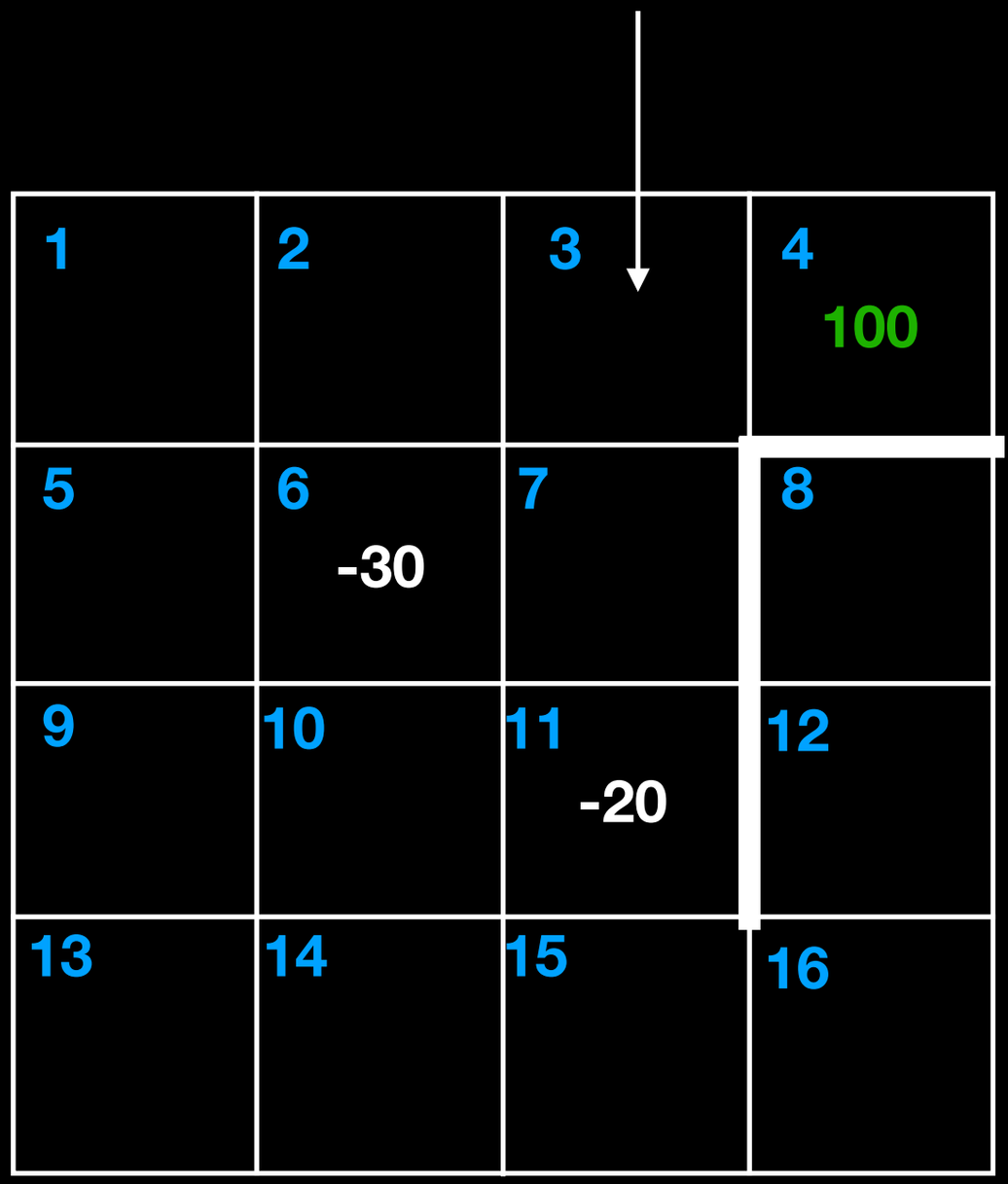
Rollout 3

Starting State = 4

Immediate Reward = 100

Return = 100

Hence, V(s = 4) = 100

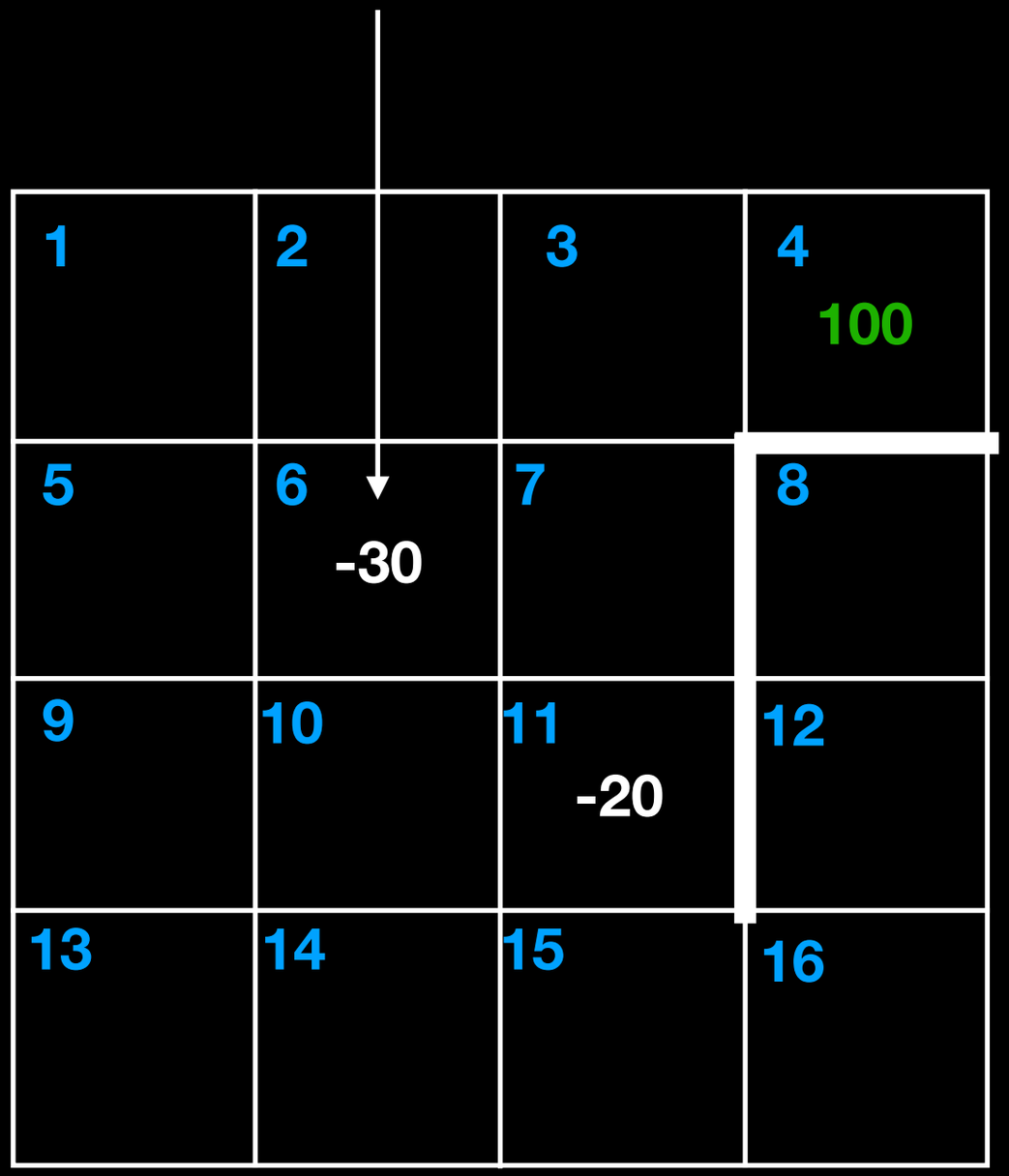


Hence, V(s=3) is
(99 + 65 + ….) / number of rollouts

Rollout 1
Starting State = 3
Immediate Reward = 0
Transition to State = 4
Transition Reward = -1
Immediate Reward = 100
Return = 99

Rollout 2
Starting State = 3
Immediate Reward = 0
Transition to State = 7
Transition Reward = -1
Immediate Reward = 0
Transition to State = 6
Transition Reward = -1
Immediate Reward = -30
Transition to State = 2
Transition Reward = -1
Immediate Reward = 0
Transition to State = 3
Transition Reward = -1
Immediate Reward = 0
Transition to State = 4
Transition Reward = -1
Immediate Reward = 100
Return = 65, etc.

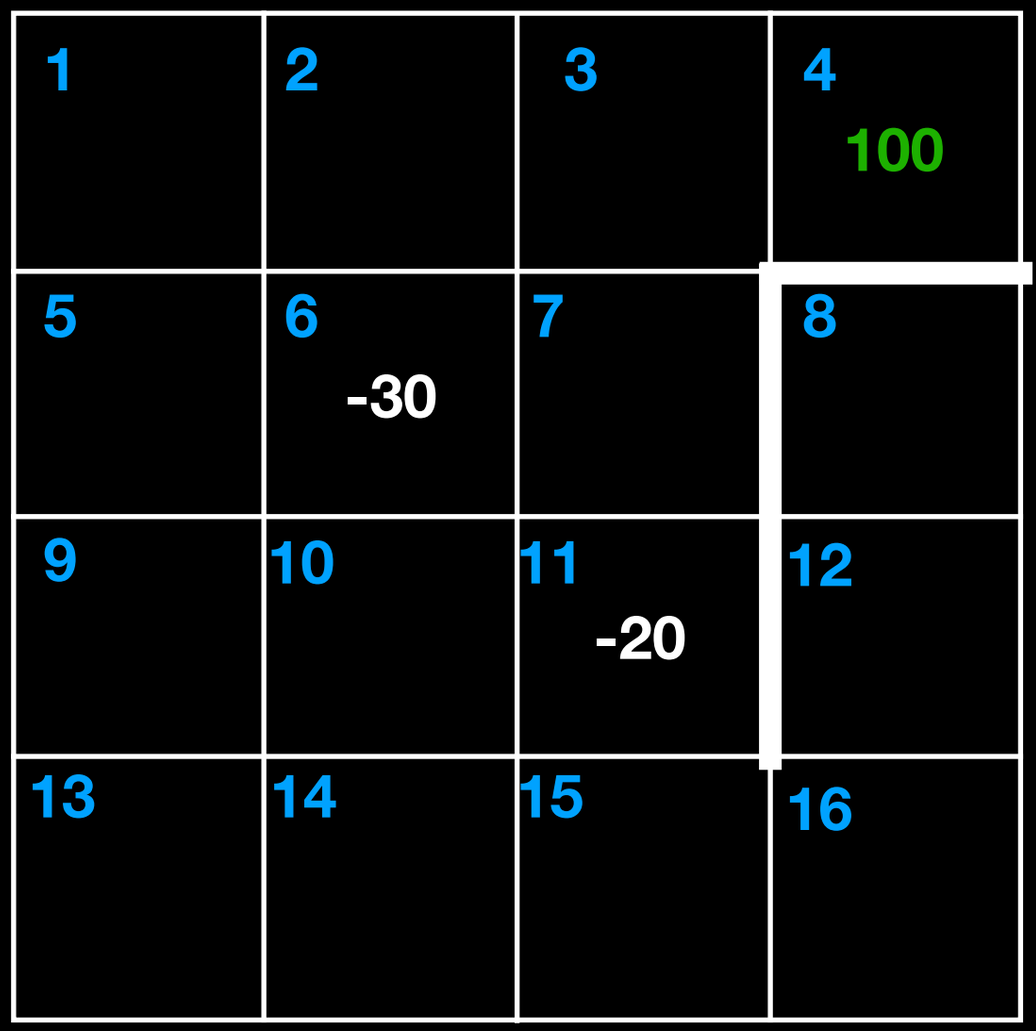
Starting State



Hence, V(s=6) is
(67 + 67 + ….) / number of rollouts

Rollout 1
Starting State = 6
Immediate Reward = -30
Transition to State = 7
Transition Reward = -1
Immediate Reward = 0
Transition to State = 3
Transition Reward = - 1
Immediate Reward = 0
Transition to State = 4
Transition Reward = -1
Immediate Reward = 100
Return = 67

Rollout 2
Starting State = 6
Immediate Reward = -30
Transition to State = 2
Transition Reward = -1
Immediate Reward = 0
Transition to State = 3
Transition Reward = - 1
Immediate Reward = 0
Transition to State = 4
Transition Reward = -1
Immediate Reward = 100
Return = 67



Different returns are produced, depending on the starting state, number of transitions and immediate rewards.

By performing multiple rollouts for a specific starting state, the expected (= average) return can be computed for that state, for a given policy.

When repeated for every possible state, we obtain the value function $V_\pi(s)$ for the given policy.