Markov Decision Processes

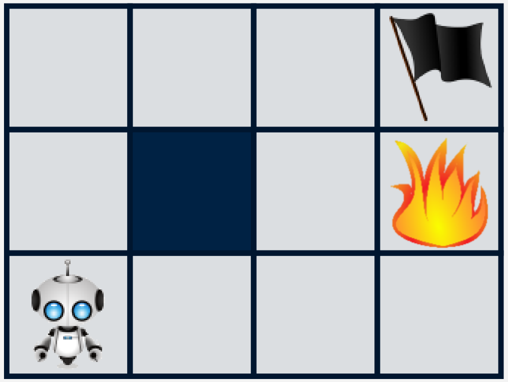
Stephen G. Ware

CSCI 4525 / 5525

Narrative Intelligence LAB

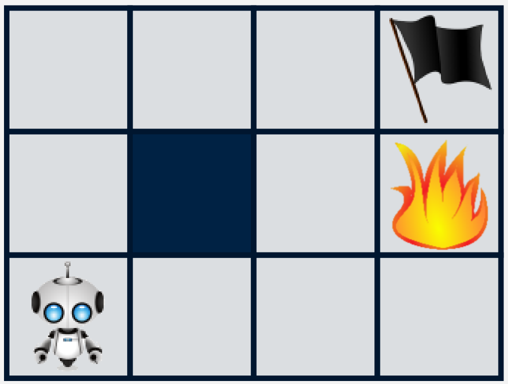
THE UNIVERSITY of NEW ORLEANS

Grid World



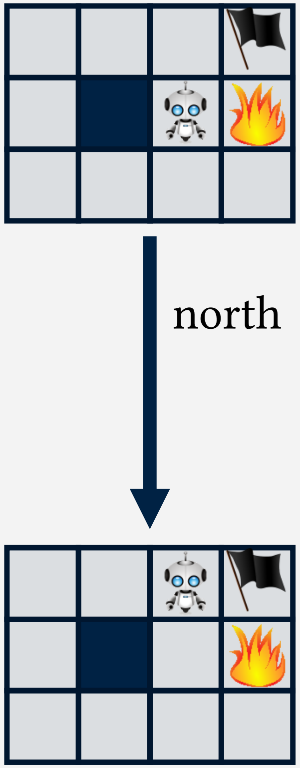
- Map has walls, fires, and a goal.
- Robot can move N, S, E, W.

Grid World



Observable:	Fully
Agents:	Single
Deterministic:	Deterministic
Episodic:	Sequetial
Static:	Static
Discrete:	Discrete

Grid World State Space



Deterministic Process

A **deterministic process** describes how the world transitions from one state to another by taking actions. It can be described as a graph whose nodes are states and whose edges are actions.

Most state spaces that we have considered so far are a deterministic processes because there is no uncertainty about the outcomes of an action.

Deterministic Decision Process

A **deterministic decision process** is defined as:

- A set of states $s \in S$
- A set of actions $a \in A$
- A start state s_0
- Optionally a set of terminal states $\{g_1, g_2 \ldots\} \in S$
- A reward function $R(s, a, s')$

Deterministic Decision Process

A **deterministic decision process** is defined as:

- A set of states $s \in S$
- A set of actions $a \in A$
- A start state s_0
- Optionally a set of terminal states $\{g_1, g_2 \dots\} \in S$
- A reward function $R(s, a, s')$

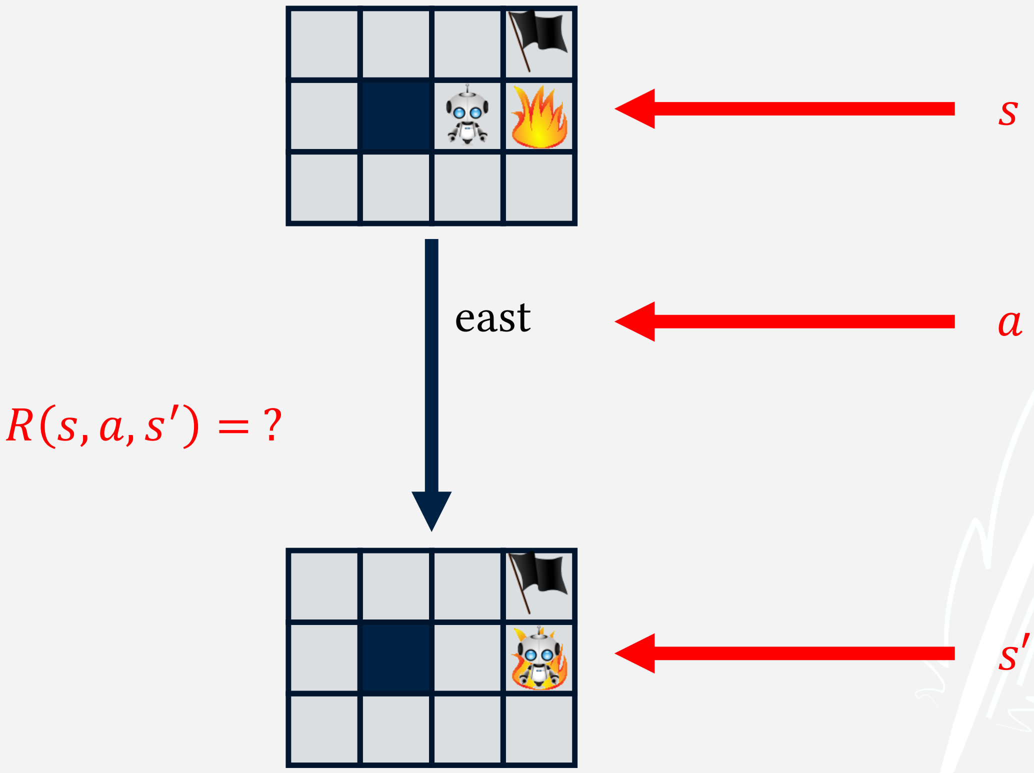
If you are in state s and you take action a to get to state s' how good or bad is it?

NIL

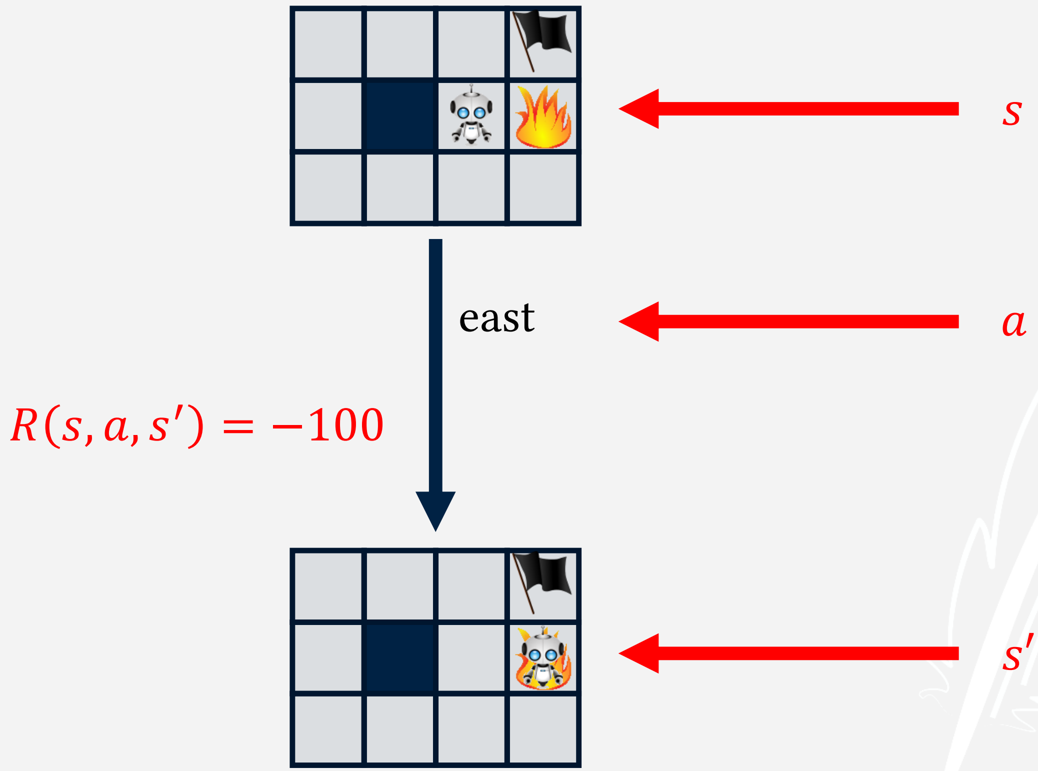
Rewards

- Rather than specifying a goal, a decision process specifies rewards.
- Rewards can be positive.
- Rewards can be negative (i.e. punishment).
- A rational agent should try to maximize its reward.
- A decision process can go on forever.
- The idea of a “goal” can be expressed as reaching a terminal state after maximizing your reward.

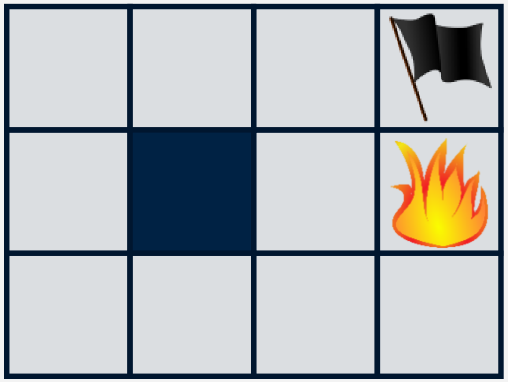
Grid World Rewards



Grid World Rewards

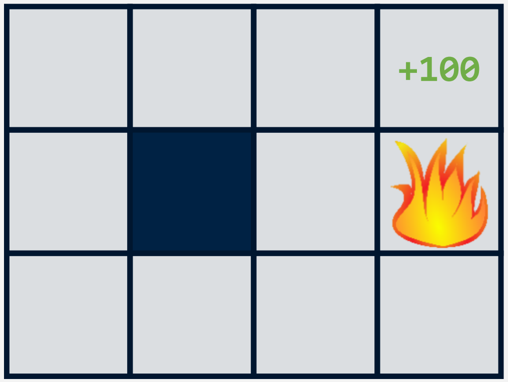


Designing Rewards



Rewards based on what we want:

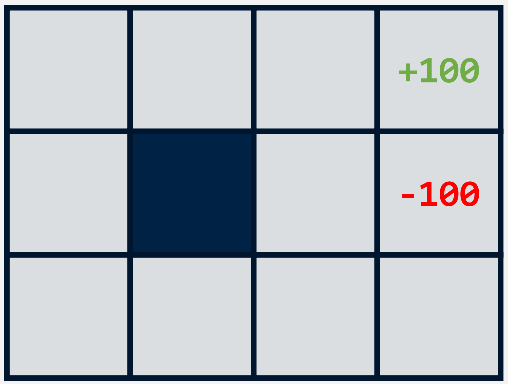
Designing Rewards



Rewards based on what we want:

- Going to the flag is good.

Designing Rewards



Rewards based on what we want:

- Going to the flag is good.
- Going to the fire is bad.

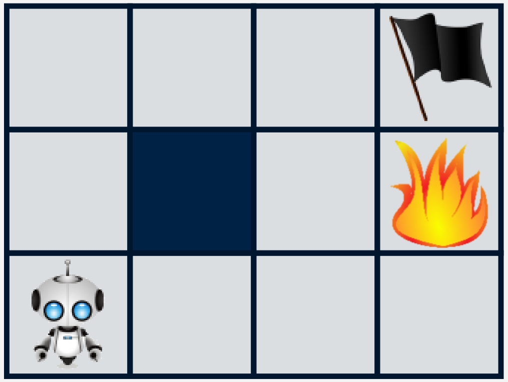
Designing Rewards

-1	-1	-1	+100
-1		-1	-100
-1	-1	-1	-1

Rewards based on what we want:

- Going to the flag is good.
- Going to the fire is bad.
- Faster is better than slower.

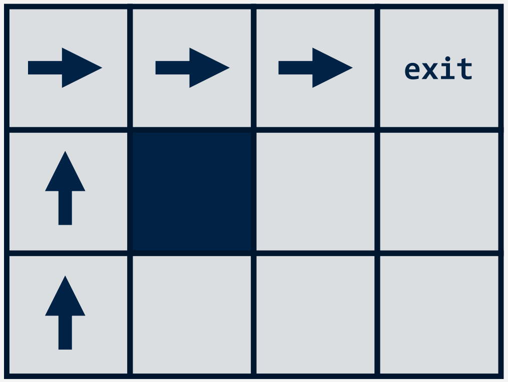
Policy



The solution to a decision process is called a **policy**.

A policy tells us what to do in every state we might be in.

Policy



The solution to a decision process is called a **policy**.

A policy tells us what to do in every state we might be in.

For a deterministic decision process, a policy is simply a **plan**.

Markov Process (Markov Chain)

When the probability of transitioning to a next state depends only on the previous state and the action taken, we say a process has the **Markov Property**.

This property is named for Andrey Markov, a famous mathematician who studied stochastic processes.

Markov Process

In other words, we don’t know the past states you have been in or the past actions you have taken.

We only know your current state and the action you intend to take. From that, we can predict (stochastically) which next state you will be in after taking the action.

Markov Decision Process

A **Markov decision process** is defined as:

- A set of states $s \in S$
- A set of actions $a \in A$
- A start state s_0
- Optionally a set of terminal states $\{g_1, g_2 \dots\} \in S$
- A reward function $R(s, a, s')$
- A transition function $T(s, a, s')$

Markov Decision Process

A **Markov decision process** is defined as:

- A set of states $s \in S$
- A set of actions $a \in A$
- A start state s_0
- Optionally a set of terminal states $\{g_1, g_2 \ldots\} \in S$
- A reward function $R(s, a, s')$
- A transition function $T(s, a, s')$

Same as for a deterministic process.

Markov Decision Process

A **Markov decision process** is defined as:

- A set of states $s \in S$
- A set of actions $a \in A$
- A start state s_0
- Optionally a set of terminal states $\{g_1, g_2 \ldots\} \in S$
- A reward function $R(s, a, s')$
- A transition function $T(s, a, s') = P(s'|s, a)$

What is the probability that if you are here…

Markov Decision Process

A **Markov decision process** is defined as:

- A set of states $s \in S$
- A set of actions $a \in A$
- A start state s_0
- Optionally a set of terminal states $\{g_1, g_2 \ldots\} \in S$
- A reward function $R(s, a, s')$
- A transition function $T(s, a, s') = P(s'|s, a)$

… and you do this…

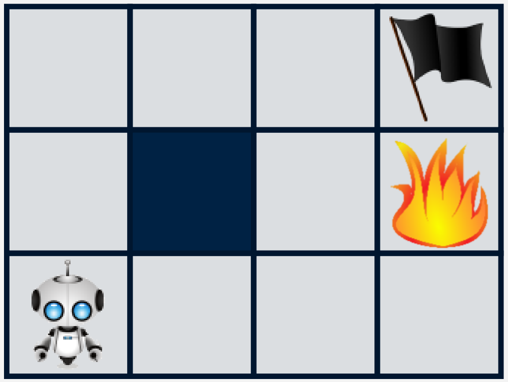
Markov Decision Process

A **Markov decision process** is defined as:

- A set of states $s \in S$
- A set of actions $a \in A$
- A start state s_0
- Optionally a set of terminal states $\{g_1, g_2 \ldots\} \in S$
- A reward function $R(s, a, s')$
- A transition function $T(s, a, s') = P(s'|s, a)$

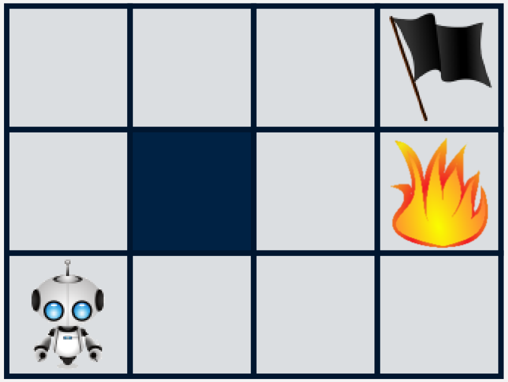
… you will end up here?

Grid World



- Map has walls, fires, and a goal.
- Robot can move N, S, E, W.

Stochastic Grid World

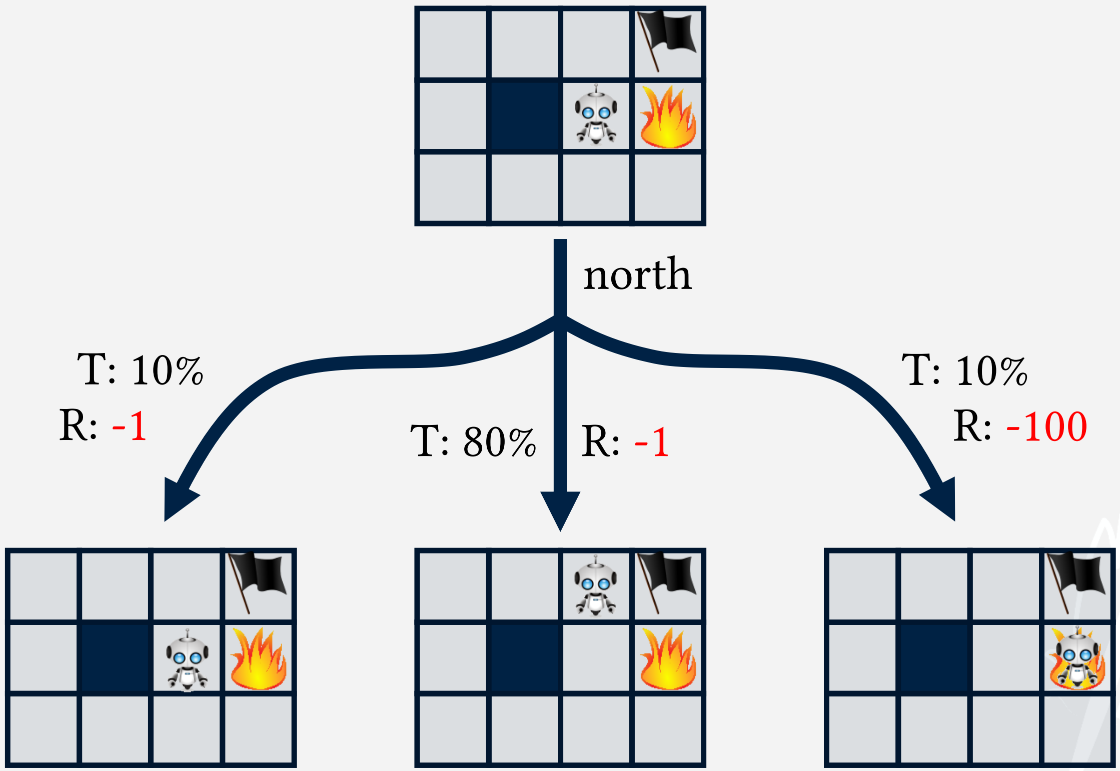


- Map has walls, fires, and a goal.
- Faulty robot can move N, S, E, W.
- When moving, there is a 10% chance you will veer to the side.

When moving N:

- $P(N) = 80\%$
- $P(E) = 10\%$
- $P(W) = 10\%$

Stochastic Grid World



Stochastic Grid World Policy

Rewards

-1	-1	-1	+100
-1		-1	-100
-1	-1	-1	-1

Policy

			exit
			exit

Stochastic Grid World Policy

Rewards

-1	-1	-1	+100
-1		-1	-100
-1	-1	-1	-1

Policy

→	→	→	exit
↑		←	exit
↑	←	←	↓

NIL

Stochastic Grid World Policy

Rewards

-3	-3	-3	+100
-3		-3	-100
-3	-3	-3	-3

Policy

			exit
			exit

NIL

Stochastic Grid World Policy

Rewards

-3	-3	-3	+100
-3		-3	-100
-3	-3	-3	-3

Policy

→	→	→	exit
↑		↑	exit
↑	←	←	←

Stochastic Grid World Policy

Rewards

-4	-4	-4	+100
-4		-4	-100
-4	-4	-4	-4

Policy

			exit
			exit

Stochastic Grid World Policy

Rewards

-4	-4	-4	+100
-4		-4	-100
-4	-4	-4	-4

Policy

→	→	→	exit
↑		↑	exit
↑	→	↑	←

Stochastic Grid World Policy

Rewards

-200	-200	-200	+100
-200		-200	-100
-200	-200	-200	-200

Policy

			exit
			exit

Stochastic Grid World Policy

Rewards

-200	-200	-200	+100
-200		-200	-100
-200	-200	-200	-200

Policy

→	→	→	exit
↑		→	exit
↑	→	→	↑

Reward: How much and when?

Which better?

Time 1: \$1	Time 1: \$2
Time 2: \$2	Time 2: \$3
Time 3: \$3	Time 3: \$4

Reward: How much and when?

Which better?

Time 1: $1	Time 1: $2
Time 2: $2	Time 2: $3
Time 3: $3	Time 3: $4

More is better than less.

Reward: How much and when?

Which better?

Time 1: \$1	Time 1: \$2
Time 2: \$2	Time 2: \$3
Time 3: \$3	Time 3: \$4

More is better than less.

Which better?

Time 1: \$0	Time 1: \$3
Time 2: \$0	Time 2: \$0
Time 3: \$3	Time 3: \$0

Reward: How much and when?

Which better?

Time 1: \$1	Time 1: \$2
Time 2: \$2	Time 2: \$3
Time 3: \$3	Time 3: \$4

More is better than less.

Which better?

Time 1: \$0	Time 1: \$3
Time 2: \$0	Time 2: \$0
Time 3: \$3	Time 3: \$0

Sooner is better than later.

Solving an MDP

Because this is a stochastic problem, there is no way to guarantee that we'll get the maximum reward, but we want a policy that is *most likely* to get the maximum reward.

We want to maximize *expected* reward.

Solving an MDP

$Q(s, a)$ is the expected reward if we start at state s, take action a, and then act optimally afterwards.

$V(s)$ is the expected reward if we start at state s and then act optimally.

γ is the reward discount factor.

Bellman Equation

$$Q(s,a) = \sum_{s'} T(s,a,s') \cdot \left(R(s,a,s') + \gamma V(s')\right)$$

Bellman Equation

$$Q(s,a) = \sum_{s'} T(s,a,s') \cdot \left(R(s,a,s') + \gamma V(s')\right)$$

The expected reward of taking action a from state s is…

Bellman Equation

$$Q(s,a) = \sum_{s'} T(s,a,s') \cdot \big(R(s,a,s') + \gamma V(s')\big)$$

For all possible states s' that we could end up in as a result of that action…

Bellman Equation

$$Q(s,a) = \sum_{s'} T(s,a,s') \cdot \left(R(s,a,s') + \gamma V(s')\right)$$

...the reward for ending up in that state...

Bellman Equation

$$Q(s,a) = \sum_{s'} T(s,a,s') \cdot \big(R(s,a,s') + \gamma V(s')\big)$$

...plus the expected reward afterwards...

Bellman Equation

$$Q(s,a) = \sum_{s'} T(s,a,s') \cdot \left(R(s,a,s') + \gamma V(s')\right)$$

...discounted by γ, because sooner is better than later...

NIL

Bellman Equation

$$Q(s,a) = \sum_{s'} T(s,a,s') \cdot \left(R(s,a,s') + \gamma V(s')\right)$$

…weighted by the likelihood that we will actually end up in that state.

Bellman Equation

$$Q(s,a) = \sum_{s'} T(s,a,s') \cdot \left(R(s,a,s') + \gamma V(s')\right)$$

Bellman Equation

$$Q(s,a) = \sum_{s'} T(s,a,s') \cdot \big(R(s,a,s') + \gamma V(s')\big)$$

$$V(s) = \max_a Q(s,a)$$

Bellman Equation

$$Q(s,a) = \sum_{s'} T(s,a,s') \cdot \left(R(s,a,s') + \gamma V(s')\right)$$

$$V(s) = \max_a Q(s,a)$$

The reward you should expect to get if you find yourself in state s is…

Bellman Equation

$$Q(s,a) = \sum_{s'} T(s,a,s') \cdot \left(R(s,a,s') + \gamma V(s')\right)$$

$$V(s) = \max_{a} Q(s,a)$$

...of all the possible actions a
that we could take in state s...

Bellman Equation

$$Q(s,a) = \sum_{s'} T(s,a,s') \cdot \left(R(s,a,s') + \gamma V(s')\right)$$

$$V(s) = \max_{a} Q(s,a)$$

...choose the action that maximizes expected reward.

Bellman Equation

$$Q(s,a) = \sum_{s'} T(s,a,s') \cdot \big(R(s,a,s') + \gamma V(s')\big)$$

$$V(s) = \max_{a} Q(s,a)$$

Bellman Equation

$$Q(s,a) = \sum_{s'} T(s,a,s') \cdot \big(R(s,a,s') + \gamma V(s')\big)$$

$$V(s) = \max_a Q(s,a)$$

$$V(s) = \max_a Q(s,a)$$

Bellman Equation

$$Q(s,a) = \sum_{s'} T(s,a,s') \cdot \big(R(s,a,s') + \gamma V(s')\big)$$

$$V(s) = \max_a Q(s,a)$$

$$V(s) = \max_a Q(s,a)$$

Substitute $Q(s,a)$

Bellman Equation

$$Q(s,a) = \sum_{s'} T(s,a,s') \cdot \left(R(s,a,s') + \gamma V(s')\right)$$

$$V(s) = \max_a Q(s,a)$$

$$V(s) = \max_a \sum_{s'} T(s,a,s') \cdot \left(R(s,a,s') + \gamma V(s')\right)$$

Bellman Equation

$$Q(s,a) = \sum_{s'} T(s,a,s') \cdot \left(R(s,a,s') + \gamma V(s')\right)$$

$$V(s) = \max_a Q(s,a)$$

$$V(s) = \max_a \sum_{s'} T(s,a,s') \cdot \left(R(s,a,s') + \gamma V(s')\right)$$

This is the Bellman Equation, which describes the expected reward from some given state s.

Parameters

$$V(s) = \max_a \sum_{s'} T(s,a,s') \cdot \big(R(s,a,s') + \gamma V(s')\big)$$

Consider γ, the reward discount factor.

Changing this value changes the policy.

When $\gamma = 0$:	Future rewards have no value.
When $0 < \gamma < 1$:	Sooner is better than later.
When $\gamma = 1$:	Sooner is as good as later.

Stochastic Grid World Policy

?	?	?	+100
?		?	-100
?	?	?	?

Suppose these rewards.

Suppose we are only allowed to make 0 moves.

Suppose $\gamma = 0.9$.

What is $V(s)$ for each s?

Stochastic Grid World Policy

0	0	0	+100
0		0	-100
0	0	0	0

Suppose these rewards.

Suppose we are only allowed to make 0 moves.

Suppose $\gamma = 0.9$.

What is $V(s)$ for each s?

Stochastic Grid World Policy

0	0	80	+100
0		0	-100
0	0	0	0

Suppose these rewards.

Suppose we are only allowed to make 1 moves.

Suppose $\gamma = 0.9$.

What is $V(s)$ for each s?

Stochastic Grid World Policy

0	0	80	+100
0		0	-100
0	0	0	0

Suppose these rewards.

Suppose we are only allowed to make 1 moves.

Suppose $\gamma = 0.9$.

What is $V(s)$ for each s?

Best outcome is moving east, but there's only an 80% chance it will work.

Stochastic Grid World Policy

0	?	80	+100
0		?	-100
0	0	0	0

Suppose these rewards.

Suppose we are only allowed to make 2 moves.

Suppose $\gamma = 0.9$.

What is $V(s)$ for each s?

Stochastic Grid World Policy

0	s ?	s' 80	+100
0		?	-100
0	0	0	0

$$? = T(s,a,s')\big(R(s,a,s') + \gamma V(s')\big)$$

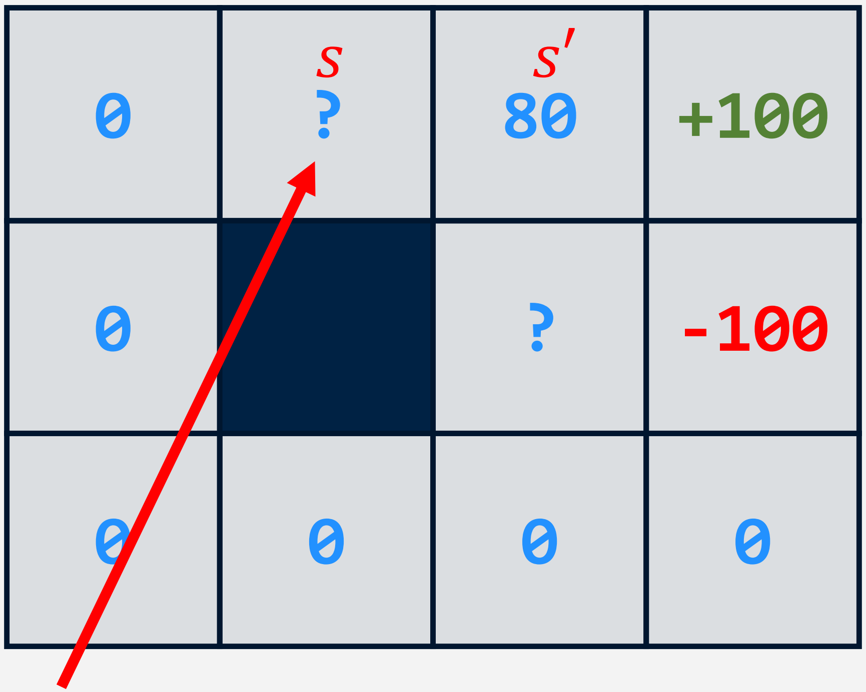
Suppose these rewards.

Suppose we are only allowed to make 2 moves.

Suppose $\gamma = 0.9$.

What is $V(s)$ for each s?

Stochastic Grid World Policy



Suppose these rewards.

Suppose we are only allowed to make 2 moves.

Suppose $\gamma = 0.9$.

What is $V(s)$ for each s?

$$? = T(s, east, s')\big(R(s, east, s') + \gamma V(s')\big)$$

Stochastic Grid World Policy

0	s ?	s' 80	+100
0		?	-100
0	0	0	0

$$? = 0.8\big(R(s, east, s') + \gamma V(s')\big)$$

Suppose these rewards.

Suppose we are only allowed to make 2 moves.

Suppose $\gamma = 0.9$.

What is $V(s)$ for each s?

Stochastic Grid World Policy

0	s ?	s' 80	+100
0		?	-100
0	0	0	0

$$? = 0.8\big(0 + \gamma V(s')\big)$$

Suppose these rewards.

Suppose we are only allowed to make 2 moves.

Suppose $\gamma = 0.9$.

What is $V(s)$ for each s?

Stochastic Grid World Policy

0	s ?	s' 80	+100
0		?	-100
0	0	0	0

$$? = 0.8\big(0 + 0.9 \cdot V(s')\big)$$

Suppose these rewards.

Suppose we are only allowed to make 2 moves.

Suppose $\gamma = 0.9$.

What is $V(s)$ for each s?

Stochastic Grid World Policy

0	s ?	s' 80	+100
0		?	-100
0	0	0	0

$? = 0.8(0 + 0.9 \cdot 80)$

Suppose these rewards.

Suppose we are only allowed to make 2 moves.

Suppose $\gamma = 0.9$.

What is $V(s)$ for each s?

Stochastic Grid World Policy

0	57	87	+100
0		47	-100
0	0	0	0

Suppose these rewards.

Suppose we are only allowed to make 2 moves.

Suppose $\gamma = 0.9$.

What is $V(s)$ for each s?

Stochastic Grid World Policy

41	73	92	+100
0		57	-100
0	0	34	0

Suppose these rewards.

Suppose we are only allowed to make 3 moves.

Suppose $\gamma = 0.9$.

What is $V(s)$ for each s?

Stochastic Grid World Policy

56	79	93	+100
29		61	-100
0	24	41	14

Suppose these rewards.

Suppose we are only allowed to make 4 moves.

Suppose $\gamma = 0.9$.

What is $V(s)$ for each s?

Stochastic Grid World Policy

65	81	93	+100
45		62	-100
23	34	47	20

Suppose these rewards.

Suppose we are only allowed to make 5 moves.

Suppose $\gamma = 0.9$.

What is $V(s)$ for each s?

Stochastic Grid World Policy

68	82	94	+100
55		63	-100
38	40	50	26

Suppose these rewards.

Suppose we are only allowed to make 6 moves.

Suppose $\gamma = 0.9$.

What is $V(s)$ for each s?

Stochastic Grid World Policy

?	?	?	+100
?		?	-100
?	?	?	?

Suppose these rewards.

Suppose we are allowed to make ∞ moves.

Suppose $\gamma = 0.9$.

What is $V(s)$ for each s?

Value Iteration

If the Markov process is known, we can find an optimal policy by calculating the expected reward for 1 move, then 2 moves, then 3 moves, etc. until the values stop changing.

Even if the Markov process never ends, when $\gamma < 1$, the values eventually converge, because future rewards become so heavily discounted they eventually approach 0.

But this is very inefficient!

Markov Decision Process

A **Markov decision process** is defined as:

- A set of states $s \in S$
- A set of actions $a \in A$
- A start state s_0
- Optionally a set of terminal states $\{g_1, g_2 \ldots\} \in S$
- A reward function $R(s, a, s')$
- A transition function $T(s, a, s')$

Markov Decision Process

A **Markov decision process** is defined as:

- A set of states $s \in S$
- A set of actions $a \in A$
- A start state s_0
- Optionally a set of terminal states
- A reward function $R(s, a, s')$
- A transition function $T(s, a, s')$

Value iteration and other direct solutions require all these things to be known in advance.

Markov Decision Process

A **Markov decision process** is defined as:

- A set of states $s \in S$
- A set of actions $a \in A$
- A start state s_0
- Optionally a set of terminal states
- A reward function $R(s, a, s')$
- A transition function $T(s, a, s')$

What if these things are fixed (i.e. do not change) but unknown?

Reinforcement Learning

- We assume the world works like a stochastic process that obeys the Markov property.
- We will explore the world, learning about new states and their rewards as we go.
- Over time, we will learn which states lead to high rewards and which lead to low.

Exploring an MDP

```
To explore an unknown MDP that starts in state s0:
Until you run out of time to explore:
   Let s be s0.
   Until s is a terminal state:
      Choose action a at random.
      Let s' be the state after taking a in s.
      Let s be s'.
```

Exploiting an MDP

```
To exploit an unknown MDP that starts in state s0:
Until you run out of time to explore:
   Let s be s0.
   Until s is a terminal state:
      Choose action a that has highest expected reward.
      Let s' be the state after taking a in s.
      Let s be s'.
```

Exploiting an MDP

```
To exploit an unknown MDP that starts in state s0:
Until you run out of time to explore:
    Let s be s0.
    Until s is a terminal state:
        Choose action a that has highest expected reward.
        Let s' be the state after taking a in s.
        Let s be s'.
```

Explore and Exploit an MDP

```
To learn an unknown MDP that starts in state s0:
Let 0 ≤ n ≤ 1 be the "noise" parameter.
Until you run out of time to explore:
   Let s be s0.
   Until s is a terminal state:
      With probability < n:
         Choose action a at random.
      Else:
         Choose action a that has highest expected reward.
      Let s' be the state after taking a in s.
      Let s be s'.
```

Q Table

State	north	south	east	west
$x = 1, y = 1$	0.95	0.55	0.05	-0.35
$x = 2, y = 1$	0.62	-0.25	0.50	-0.20
…	…	…	…	…
$x = 5, y = 5$	0.23	-0.04	0.52	0.18

Q Table

State	north	south	east	west
$x = 1, y = 1$	0.95	0.55	0.05	-0.35
$x = 2, y = 1$	0.62	-0.25	0.50	-0.20
…	…	…	…	…
$x = 5, y = 5$	0.23	-0.04	0.52	0.18

One row for each state.

NIL

Q Table

State	north	south	east	west
$x = 1, y = 1$	0.95	0.55	0.05	0.35
$x = 2, y = 1$	0.62	-0.25	0.50	-0.20
…	…	…	…	…
$x = 5, y = 5$	0.23	-0.04	0.52	0.18

One column for each action.

Q Table

State	north	south	east	west
$x = 1, y = 1$	0.95	0.55	0.05	-0.35
$x = 2, y = 1$	0.62	-0.25	0.50	-0.20
…	…	…	…	…
$x = 5, y = 5$	0.23	-0.04	0.52	0.18

An individual cell represents the expected reward for taking some action in some state.

Q Table

State	north	south	east	west
$x = 1, y = 1$	0.95	0.55	0.05	-0.35
$x = 2, y = 1$	0.62	-0.25	0.50	-0.20
…	…	…	…	…
$x = 5, y = 5$	0.23	-0.04	0.52	0.18

The name "Q Table" comes from this equation:

$$Q(s,a) = \sum_{s'} T(s,a,s') \cdot \big(R(s,a,s') + \gamma V(s')\big)$$

Q Table

State	north	south	east	west
$x = 1, y = 1$	0.95	0.55	0.05	-0.35
$x = 2, y = 1$	0.62	-0.25	0.50	-0.20
…	…	…	…	…
$x = 5, y = 5$	0.23	-0.04	0.52	0.18

You can use this table to calculate:

$$V(s) = \max_a Q(s, a)$$

Q Table

State	north	south	east	west
$x = 1, y = 1$	0.95	0.55	0.05	-0.35
$x = 2, y = 1$	0.62	0.25	0.50	0.20
…	…	…	…	…
$x = 5, y = 5$	0.23	-0.04	0.52	0.18

What's the best outcome in this state?

You can use this table to calculate:

$$V(s) = \max_a Q(s,a)$$

Q Table

State	north	south	east	west
$x = 1, y = 1$	0.95	0.55	0.05	-0.35
$x = 2, y = 1$	0.62	-0.25	0.50	-0.20
…	…	…	…	…
$x = 5, y = 5$	0.23	-0.04	0.52	0.18

This is the expected reward.

You can use this table to calculate:

$$V(s) = \max_{a} Q(s, a)$$

Q Table

State	north	south	east	west
$x = 1, y = 1$	0.95	0.55	0.05	-0.35
$x = 2, y = 1$	0.62	-0.25	0.50	-0.20
…	…	…	…	…
$x = 5, y = 5$	0.23	-0.04	0.52	0.18

This is the action you should take.

You can use this table to calculate:

$$V(s) = \max_{a} Q(s, a)$$

Q Learning

```
Let 0 ≤ n ≤ 1 be the "noise" parameter.

Let 0 ≤ α ≤ 1 be the "learning rate" parameter.

Let 0 ≤ γ ≤ 1 be the "reward discount" parameter.

Let Q(s,a) be a table with a row for each state s and a
column for each action a.

Let V(s) be a function which returns the highest value
of Q(s,a) for all a.

Initially, for all s and for all a, Q(s,a) = 0.

...
```

Q Learning

```
...
Until you run out of time to explore:
   Let s be s0.
   Until s is a terminal state:
      With probability < n:
         Choose action a at random.
      Else:
         Choose action a that maximizes Q(s,a).
      Let s' be the state after taking a in s.
      Let r be the reward for taking a in s.
      Let Q(s,a) = (1 - α) Q(s,a) + α(r + γV(s'))
      Let s be s'.
```

Q Learning Parameters

```
With probability < n move randomly.
```

n is the noise.

- Low values mean the agent rarely makes random moves (prefers to exploit).
- High values mean the agent often makes random moves (prefers to explore).

Q Learning Parameters

$$Q(s,a) = (1 - \alpha)\, Q(s,a) + \alpha(r + \gamma V(s'))$$

α is the learning rate.

- Low values mean the agent overwrites old values in the Q table quickly (forgetful agent).
- High values mean the agent overwrites old values in the Q table slowly (an agent set in its ways).

Q Learning Parameters

$$Q(s,a) = (1 - \alpha)\ Q(s,a) + \alpha(r + \gamma V(s'))$$

γ is the discount factor.

- Low values mean the agent cares more about short-term rewards (shortsighted or myopic agent).
- High values mean the agent cares more about long-term rewards (farsighted agent).

Q Learning Policies

- Typically, reinforcement learning agents alternate between two phases: learning and evaluation.
- Learning is when you develop your policy.
- During learning, explore and exploit and update the Q Table as you go.
- Evaluation is when you test your policy.
- During evaluation, always exploit (i.e. do the action you think will be get you the highest score).

Q Learning Policies

- A *policy* is a function which takes a state as input and returns the action an agent should do.
- During learning, a Q Learning agent fills in its Q Table.
- Policy: In state s, check row s in the Q Table and choose the action with the highest value.
- In other words, in state s, always choose the action a that maximizes $Q(s, a)$.