Trajectory Optimization

CISC 7404 - Decision Making

Steven Morad

University of Macau

Exam

Review

MDP Coding

Decision Making with a Model

Decision Making with a Model

Decision Making with a Model

In RL, our goal is to optimize the discounted return

Decision Making with a Model

In RL, our goal is to optimize the discounted return

Today, we will see some methods to do this

Decision Making with a Model

In RL, our goal is to optimize the discounted return

Today, we will see some methods to do this

These ideas are very old, and do not necessarily require deep learning

Decision Making with a Model

In RL, our goal is to optimize the discounted return

Today, we will see some methods to do this

These ideas are very old, and do not necessarily require deep learning

Many of these ideas appear in classical robotics and control theory

Decision Making with a Model

In RL, our goal is to optimize the discounted return

Today, we will see some methods to do this

These ideas are very old, and do not necessarily require deep learning

Many of these ideas appear in classical robotics and control theory

These methods are expensive in terms of compute

Decision Making with a Model

In RL, our goal is to optimize the discounted return

Today, we will see some methods to do this

These ideas are very old, and do not necessarily require deep learning

Many of these ideas appear in classical robotics and control theory

These methods are expensive in terms of compute

We usually only use these methods for simple problems

Decision Making with a Model

“Simple” problems have small state and actions spaces

$$|S|, |A| = \text{small}$$

Decision Making with a Model

“Simple” problems have small state and actions spaces

$$|S|, |A| = \text{small}$$

One example is position and velocity control

Decision Making with a Model

“Simple” problems have small state and actions spaces

$$|S|, |A| = \text{small}$$

One example is position and velocity control

https://www.youtube.com/watch?v=6qj3EfRTtkE

Decision Making with a Model

“Simple” problems have small state and actions spaces

$$|S|, |A| = \text{small}$$

One example is position and velocity control

https://www.youtube.com/watch?v=6qj3EfRTtkE

Decision Making with a Model

Given the power of modern GPUs, researchers are revisiting these methods

Decision Making with a Model

Given the power of modern GPUs, researchers are revisiting these methods

They are applying them to more difficult tasks with large $|S|, |A|$

Decision Making with a Model

Given the power of modern GPUs, researchers are revisiting these methods

They are applying them to more difficult tasks with large $|S|, |A|$

https://youtu.be/_e3BKzK6xD0?si=Kr-KOccTDypgRjgJ&t=194

Decision Making with a Model

There are two classes of decision making algorithms

Decision Making with a Model

There are two classes of decision making algorithms

Model-based

Decision Making with a Model

There are two classes of decision making algorithms

Model-based

We know $\mathrm{Tr}(s_{t+1} \mid s_t, a_t)$

Decision Making with a Model

There are two classes of decision making algorithms

Model-based

We know $\mathrm{Tr}(s_{t+1} \mid s_t, a_t)$

Cheap to train, expensive to use

Decision Making with a Model

There are two classes of decision making algorithms

Model-based

We know $\mathrm{Tr}(s_{t+1} \mid s_t, a_t)$

Cheap to train, expensive to use

Closer to traditional control theory

Decision Making with a Model

There are two classes of decision making algorithms

Model-based

We know $\mathrm{Tr}(s_{t+1} \mid s_t, a_t)$

Cheap to train, expensive to use

Closer to traditional control theory

Model-free

Decision Making with a Model

There are two classes of decision making algorithms

Model-based

We know $\mathrm{Tr}(s_{t+1} \mid s_t, a_t)$

Cheap to train, expensive to use

Closer to traditional control theory

Model-free

We do not know $\mathrm{Tr}(s_{t+1} \mid s_t, a_t)$

Decision Making with a Model

There are two classes of decision making algorithms

Model-based

We know $\mathrm{Tr}(s_{t+1} \mid s_t, a_t)$

Cheap to train, expensive to use

Closer to traditional control theory

Model-free

We do not know $\mathrm{Tr}(s_{t+1} \mid s_t, a_t)$

Expensive to train, cheap to use

Decision Making with a Model

There are two classes of decision making algorithms

Model-based

We know $\mathrm{Tr}(s_{t+1} \mid s_t, a_t)$

Cheap to train, expensive to use

Closer to traditional control theory

Model-free

We do not know $\mathrm{Tr}(s_{t+1} \mid s_t, a_t)$

Expensive to train, cheap to use

Closer to deep learning

Today, we will cover a model-based algorithm called trajectory optimization

Decision Making with a Model

There are two classes of decision making algorithms

Model-based

We know $\mathrm{Tr}(s_{t+1} \mid s_t, a_t)$

Cheap to train, expensive to use

Closer to traditional control theory

Model-free

We do not know $\mathrm{Tr}(s_{t+1} \mid s_t, a_t)$

Expensive to train, cheap to use

Closer to deep learning

Today, we will cover a model-based algorithm called trajectory optimization

Critical part of Alpha-* methods (AlphaGo, AlphaStar, AlphaZero)

Decision Making with a Model

Recall the discounted return, our objective for the rest of this course

Decision Making with a Model

Recall the discounted return, our objective for the rest of this course

$$G(\boldsymbol{\tau}) = \sum_{t=0}^{\infty} \gamma^t R(s_{t+1})$$

Decision Making with a Model

Recall the discounted return, our objective for the rest of this course

$$G(\boldsymbol{\tau}) = \sum_{t=0}^{\infty} \gamma^t R(s_{t+1})$$

We want to maximize the discounted return

$$\arg \max_{\boldsymbol{\tau}} G(\boldsymbol{\tau}) = \arg \max_{s \in S} \sum_{t=0}^{\infty} \gamma^t R(s_{t+1})$$

Decision Making with a Model

Recall the discounted return, our objective for the rest of this course

$$G(\boldsymbol{\tau}) = \sum_{t=0}^{\infty} \gamma^t R(s_{t+1})$$

We want to maximize the discounted return

$$\arg\max_{\boldsymbol{\tau}} G(\boldsymbol{\tau}) = \arg\max_{s \in S} \sum_{t=0}^{\infty} \gamma^t R(s_{t+1})$$

We want to find the trajectory $\tau = \begin{bmatrix} s_0 & a_0 \\ s_1 & a_1 \\ \vdots & \vdots \end{bmatrix}$ that provides the greatest discounted return

Decision Making with a Model

$$\arg\max_{\boldsymbol{\tau}} G(\boldsymbol{\tau}) = \arg\max_{s \in S} \sum_{t=0}^{\infty} \gamma^t R(s_{t+1})$$

Decision Making with a Model

$$\arg\max_{\boldsymbol{\tau}} G(\boldsymbol{\tau}) = \arg\max_{s \in S} \sum_{t=0}^{\infty} \gamma^t R(s_{t+1})$$

This objective looks simple, but $R(s_{t+1})$ hides much of the process

Decision Making with a Model

$$\arg\max_{\boldsymbol{\tau}} G(\boldsymbol{\tau}) = \arg\max_{s \in S} \sum_{t=0}^{\infty} \gamma^t R(s_{t+1})$$

This objective looks simple, but $R(s_{t+1})$ hides much of the process

To understand what is hiding, let us examine the reward function

The Mysterious Reward

The Mysterious Reward

Consider the reward function

$$R(s_{t+1})$$

The Mysterious Reward

Consider the reward function

$$R(s_{t+1})$$

Perhaps we want to maximize the reward

$$\arg\max_{s_{t+1} \in S} R(s_{t+1})$$

The Mysterious Reward

Consider the reward function

$$R(s_{t+1})$$

Perhaps we want to maximize the reward

$$\arg\max_{s_{t+1} \in S} R(s_{t+1})$$

Question: Agent in a state s_t takes action a_t, what is $R(s_{t+1})$?

The Mysterious Reward

Consider the reward function

$$R(s_{t+1})$$

Perhaps we want to maximize the reward

$$\arg\max_{s_{t+1} \in S} R(s_{t+1})$$

Question: Agent in a state s_t takes action a_t, what is $R(s_{t+1})$?

Answer: Not sure. $R(s_{t+1})$ depends on $\mathrm{Tr}(s_{t+1} \mid s_t, a_t)$

The Mysterious Reward

Consider the reward function

$$R(s_{t+1})$$

Perhaps we want to maximize the reward

$$\arg\max_{s_{t+1} \in S} R(s_{t+1})$$

Question: Agent in a state s_t takes action a_t, what is $R(s_{t+1})$?

Answer: Not sure. $R(s_{t+1})$ depends on $\mathrm{Tr}(s_{t+1} \mid s_t, a_t)$

Cannot know s_{t+1} with certainty, only know the distribution!

The Mysterious Reward

s_{t+1} is the **outcome** of a random process

The Mysterious Reward

s_{t+1} is the **outcome** of a random process

$$s_{t+1} \sim \mathrm{Tr}(\cdot \mid s_t, a_t), \quad s_t, s_{t+1} \in S$$

The Mysterious Reward

s_{t+1} is the **outcome** of a random process

$$s_{t+1} \sim \mathrm{Tr}(\cdot \mid s_t, a_t), \quad s_t, s_{t+1} \in S$$

Question: What is S?

Answer: State space, also the outcome space Ω of Tr

$$s_{t+1} \in S = \omega \in \Omega$$

And the reward function is a scalar function of the outcome

The Mysterious Reward

s_{t+1} is the **outcome** of a random process

$$s_{t+1} \sim \mathrm{Tr}(\cdot \mid s_t, a_t), \quad s_t, s_{t+1} \in S$$

Question: What is S?

Answer: State space, also the outcome space Ω of Tr

$$s_{t+1} \in S = \omega \in \Omega$$

And the reward function is a scalar function of the outcome

$$R : S \mapsto \mathbb{R}$$

The Mysterious Reward

If you can answer the following question, you understand the course

The Mysterious Reward

If you can answer the following question, you understand the course

$$s_{t+1} \sim \mathrm{Tr}(\cdot \mid s_t, a_t), \quad s_t, s_{t+1} \in S$$

$$R : S \mapsto \mathbb{R}$$

Question: R is a special kind of function, what is it?

The Mysterious Reward

If you can answer the following question, you understand the course

$$s_{t+1} \sim \mathrm{Tr}(\cdot \mid s_t, a_t), \quad s_t, s_{t+1} \in S$$

$$R : S \mapsto \mathbb{R}$$

Question: R is a special kind of function, what is it?

Answer: R is a random variable!

The Mysterious Reward

If you can answer the following question, you understand the course

$$s_{t+1} \sim \mathrm{Tr}(\cdot \mid s_t, a_t), \quad s_t, s_{t+1} \in S$$

$$R : S \mapsto \mathbb{R}$$

Question: R is a special kind of function, what is it?

Answer: R is a random variable!

$R : S \mapsto \mathbb{R}$ $\quad\quad$ $S = \Omega$ $\quad\quad$ $R : \Omega \mapsto \mathbb{R}$

We should write it as $\mathcal{R} : S \mapsto \mathbb{R}$

The Mysterious Reward

$$\mathcal{R} : S \mapsto \mathbb{R}$$

Question: What do we like to do with random variables?

The Mysterious Reward

$$\mathcal{R} : S \mapsto \mathbb{R}$$

Question: What do we like to do with random variables?

Answer: Take the expectation!

The Mysterious Reward

$$\mathcal{R} : S \mapsto \mathbb{R}$$

Question: What do we like to do with random variables?

Answer: Take the expectation!

We cannot know which reward we get in the future

$$\mathcal{R}(s_{t+1}), \quad s_{t+1} \sim \mathrm{Tr}(\cdot \mid s_t, a_t)$$

The Mysterious Reward

$$\mathcal{R} : S \mapsto \mathbb{R}$$

Question: What do we like to do with random variables?

Answer: Take the expectation!

We cannot know which reward we get in the future

$$\mathcal{R}(s_{t+1}), \quad s_{t+1} \sim \mathrm{Tr}(\cdot \mid s_t, a_t)$$

But we can know the **average** future reward using the expectation

The Mysterious Reward

$$\mathcal{R} : S \mapsto \mathbb{R}$$

Question: What do we like to do with random variables?

Answer: Take the expectation!

We cannot know which reward we get in the future

$$\mathcal{R}(s_{t+1}), \quad s_{t+1} \sim \mathrm{Tr}(\cdot \mid s_t, a_t)$$

But we can know the **average** future reward using the expectation

$$\mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_t, a_t] = \sum_{s_{t+1} \in S} \mathcal{R}(s_{t+1}) \cdot \mathrm{Tr}(s_{t+1} \mid s_t, a_t)$$

The Mysterious Reward

$$\mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_t, a_t] = \sum_{s_{t+1} \in S} \mathcal{R}(s_{t+1}) \cdot \mathrm{Tr}(s_{t+1} \mid s_t, a_t)$$

The Mysterious Reward

$$\mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_t, a_t] = \sum_{s_{t+1} \in S} \mathcal{R}(s_{t+1}) \cdot \mathrm{Tr}(s_{t+1} \mid s_t, a_t)$$

We cannot know which reward we get in the future

The Mysterious Reward

$$\mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_t, a_t] = \sum_{s_{t+1} \in S} \mathcal{R}(s_{t+1}) \cdot \mathrm{Tr}(s_{t+1} \mid s_t, a_t)$$

We cannot know which reward we get in the future

But we can know the average (expected) reward we will get

The Mysterious Reward

$$\mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_t, a_t] = \sum_{s_{t+1} \in S} \mathcal{R}(s_{t+1}) \cdot \mathrm{Tr}(s_{t+1} \mid s_t, a_t)$$

We cannot know which reward we get in the future

But we can know the average (expected) reward we will get

As an agent, we cannot directly control the world (s_t or s_{t+1})

The Mysterious Reward

$$\mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_t, a_t] = \sum_{s_{t+1} \in S} \mathcal{R}(s_{t+1}) \cdot \mathrm{Tr}(s_{t+1} \mid s_t, a_t)$$

We cannot know which reward we get in the future

But we can know the average (expected) reward we will get

As an agent, we cannot directly control the world (s_t or s_{t+1})

All we can do is choose our own action a_t

The Mysterious Reward

$$\mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_t, a_t] = \sum_{s_{t+1} \in S} \mathcal{R}(s_{t+1}) \cdot \mathrm{Tr}(s_{t+1} \mid s_t, a_t)$$

We cannot know which reward we get in the future

But we can know the average (expected) reward we will get

As an agent, we cannot directly control the world (s_t or s_{t+1})

All we can do is choose our own action a_t

Pick an action that maximizes the expected reward

The Mysterious Reward

$$\mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_t, a_t] = \sum_{s_{t+1} \in S} \mathcal{R}(s_{t+1}) \cdot \mathrm{Tr}(s_{t+1} \mid s_t, a_t)$$

We cannot know which reward we get in the future

But we can know the average (expected) reward we will get

As an agent, we cannot directly control the world (s_t or s_{t+1})

All we can do is choose our own action a_t

Pick an action that maximizes the expected reward

$$\arg\max_{a_t \in A} \mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_t, a_t] = \arg\max_{a_t \in A} \sum_{s_{t+1} \in S} \mathcal{R}(s_{t+1}) \cdot \mathrm{Tr}(s_{t+1} \mid s_t, a_t)$$

The Mysterious Reward

$$\arg\max_{a_t \in A} \mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_t, a_t] = \arg\max_{a_t \in A} \sum_{s_{t+1} \in S} \mathcal{R}(s_{t+1}) \cdot \mathrm{Tr}(s_{t+1} \mid s_t, a_t)$$

In English:

1. Compute the probability for each outcome $s \in S$, for each $a \in A$
2. Compute the reward for each possible outcome $s \in S$
3. The expected reward for $s \in S$ is probability times reward
4. Take the action $a_t \in A$ that produces the largest the expected reward

Question: Have we seen this before?

The Mysterious Reward

$$\underset{a_t \in A}{\arg\max}\, \mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_t, a_t] = \underset{a_t \in A}{\arg\max} \sum_{s_{t+1} \in S} \mathcal{R}(s_{t+1}) \cdot \mathrm{Tr}(s_{t+1} \mid s_t, a_t)$$

In English:

1. Compute the probability for each outcome $s \in S$, for each $a \in A$
2. Compute the reward for each possible outcome $s \in S$
3. The expected reward for $s \in S$ is probability times reward
4. Take the action $a_t \in A$ that produces the largest the expected reward

Question: Have we seen this before?

Answer: Bandits!

The Mysterious Reward

$$\arg\max_{a_t \in A} \mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_t, a_t] = \arg\max_{a_t \in A} \sum_{s_{t+1} \in S} \mathcal{R}(s_{t+1}) \cdot \mathrm{Tr}(s_{t+1} \mid s_t, a_t)$$

In English:

1. Compute the probability for each outcome $s \in S$, for each $a \in A$
2. Compute the reward for each possible outcome $s \in S$
3. The expected reward for $s \in S$ is probability times reward
4. Take the action $a_t \in A$ that produces the largest the expected reward

Question: Have we seen this before?

Answer: Bandits!

$$\arg\max_{a \in \{1 \dots k\}} \mathbb{E}[\mathcal{X}_a]$$

The Mysterious Reward

$$\arg\max_{a_t \in A} \mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_t, a_t] = \arg\max_{a_t \in A} \sum_{s_{t+1} \in S} \mathcal{R}(s_{t+1}) \cdot \mathrm{Tr}(s_{t+1} \mid s_t, a_t)$$

We have a name for a function that picks actions

The Mysterious Reward

$$\underset{a_t \in A}{\arg \max} \, \mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_t, a_t] = \underset{a_t \in A}{\arg \max} \sum_{s_{t+1} \in S} \mathcal{R}(s_{t+1}) \cdot \mathrm{Tr}(s_{t+1} \mid s_t, a_t)$$

We have a name for a function that picks actions

We call this the **policy**, which usually has parameters $\theta \in \Theta$

The Mysterious Reward

$$\arg\max_{a_t \in A} \mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_t, a_t] = \arg\max_{a_t \in A} \sum_{s_{t+1} \in S} \mathcal{R}(s_{t+1}) \cdot \mathrm{Tr}(s_{t+1} \mid s_t, a_t)$$

We have a name for a function that picks actions

We call this the **policy**, which usually has parameters $\theta \in \Theta$

$$\pi : S \times \Theta \mapsto \Delta A$$

The Mysterious Reward

$$\underset{a_t \in A}{\arg\max} \, \mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_t, a_t] = \underset{a_t \in A}{\arg\max} \sum_{s_{t+1} \in S} \mathcal{R}(s_{t+1}) \cdot \mathrm{Tr}(s_{t+1} \mid s_t, a_t)$$

We have a name for a function that picks actions

We call this the **policy**, which usually has parameters $\theta \in \Theta$

$$\pi : S \times \Theta \mapsto \Delta A$$

$$\pi(a_t \mid s_t; \theta) = \begin{cases} 1 \text{ if } a_t = \arg\max_{a_t \in A} \mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_t, a_t, \theta] \\ 0 \text{ otherwise} \end{cases}$$

The Mysterious Reward

$$\arg\max_{a_t \in A} \mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_t, a_t] = \arg\max_{a_t \in A} \sum_{s_{t+1} \in S} \mathcal{R}(s_{t+1}) \cdot \mathrm{Tr}(s_{t+1} \mid s_t, a_t)$$

We have a name for a function that picks actions

We call this the **policy**, which usually has parameters $\theta \in \Theta$

$$\pi : S \times \Theta \mapsto \Delta A$$

$$\pi(a_t \mid s_t; \theta) = \begin{cases} 1 \text{ if } a_t = \arg\max_{a_t \in A} \mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_t, a_t, \theta] \\ 0 \text{ otherwise} \end{cases}$$

The policy is the "brain" of the agent, it controls the agent

The Mysterious Reward

We figured out the mystery the reward function was hiding

The Mysterious Reward

We figured out the mystery the reward function was hiding

We found a policy that maximizes the reward

The Mysterious Reward

We figured out the mystery the reward function was hiding

We found a policy that maximizes the reward

We want to maximize the discounted return, not the reward!

The Mysterious Reward

We figured out the mystery the reward function was hiding

We found a policy that maximizes the reward

We want to maximize the discounted return, not the reward!

We have one last thing to do

Trajectory Optimization

Trajectory Optimization

$$\arg\max_{a_t \in A} \mathbb{E}[\mathcal{R}(s_{t+1}) \mid s_t, a_t] = \arg\max_{a_t \in A} \sum_{s_{t+1} \in S} R(s_{t+1}) \cdot \mathrm{Tr}(s_{t+1} \mid s_t, a_t)$$

What we have

$$\mathbb{E}[R(s_{t+1}) \mid s_t, a_t] = \sum_{s_{t+1} \in S} R(s_{t+1}) \cdot \Pr(s_{t+1} \mid s_t, a_t)$$

What we want

$$\mathbb{E}[G(\boldsymbol{\tau}_n) \mid s_0, a_0, a_1, ..., a_n] = ?$$

What we need

Trajectory Optimization

$$\Pr(s_t \mid s_0, a_0, a_1, a_2, ...a_{t-1})$$

Trajectory Optimization

$$\Pr(s_1 \mid s_0, a_0)$$

$$\Pr(s_2 \mid s_0, a_0, a_1) = \Pr(s_2 \mid s_1, a_1) \Pr(s_1 \mid s_0, a_0)$$

$$\begin{aligned} \Pr(s_t \mid s_0, a_0, a_1, ...a_t) &= \Pr(s_t \mid s_{t-1}) ... \Pr(s_2 \mid s_1, a_1) \Pr(s_1 \mid s_0, a_0) \\ &= \end{aligned}$$

Trajectory Optimization

$$\mathbb{E}[G(\boldsymbol{\tau}_n) \mid s_0, a_0, a_1, ..., a_n] = ?$$

Plug in definition of discounted return

$$\mathbb{E}[G(\boldsymbol{\tau}) \mid s_0, a_0, a_1, ...] = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}[R(s_{t+1})]$$

Trajectory Optimization

Goal: Given an initial state and some actions, predict the expected discounted return

Trajectory Optimization

Goal: Given an initial state and some actions, predict the expected discounted return

$$\mathbb{E}[R(s_1) \mid s_0, a_0] = \sum_{s_1 \in S} R(s_1) \Pr(s_1 \mid s_0, a_0)$$

$$\mathbb{E}[R(s_2) \mid s_0, a_0, a_1] = \sum_{s_2 \in S} R(s_2) \sum_{s_1 \in S} \Pr(s_2 \mid s_1, a_1) \Pr(s_1 \mid s_0, a_0)$$

$$\mathbb{E}[R(s_{n+1}) \mid s_0, a_0, a_1, ...a_n] = \sum_{s_{n+1} \in S} R(s_{n+1}) \sum_{s_1, ..., s_n \in S} \prod_{t=0}^{n} \Pr(s_{t+1} \mid s_t, a_t)$$

Trajectory Optimization

$$\mathbb{E}[R(s_{n+1}) \mid s_0, a_0, a_1, ...a_n] = \sum_{s_{n+1} \in S} R(s_{n+1}) \sum_{s_1, ..., s_n \in S} \prod_{t=0}^{n} \Pr(s_{t+1} \mid s_t, a_t)$$

Trajectory Optimization

$$\mathbb{E}[R(s_{n+1}) \mid s_0, a_0, a_1, \ldots a_n] = \sum_{s_{n+1} \in S} R(s_{n+1}) \sum_{s_1, \ldots, s_n \in S} \prod_{t=0}^{n} \Pr(s_{t+1} \mid s_t, a_t)$$

Mean reward over possible s_{n+1}

Trajectory Optimization

s_{n+1} Distribution

$$\mathbb{E}[R(s_{n+1}) \mid s_0, a_0, a_1, \ldots a_n] = \sum_{s_{n+1} \in S} R(s_{n+1}) \sum_{s_1, \ldots, s_n \in S} \prod_{t=0}^{n} \Pr(s_{t+1} \mid s_t, a_t)$$

Mean reward over possible s_{n+1}

Trajectory Optimization

s_{n+1} Distribution

$$\mathbb{E}[R(s_{n+1}) \mid s_0, a_0, a_1, \dots a_n] = \sum_{s_{n+1} \in S} R(s_{n+1}) \sum_{s_1, \dots, s_n \in S} \prod_{t=0}^{n} \Pr(s_{t+1} \mid s_t, a_t)$$

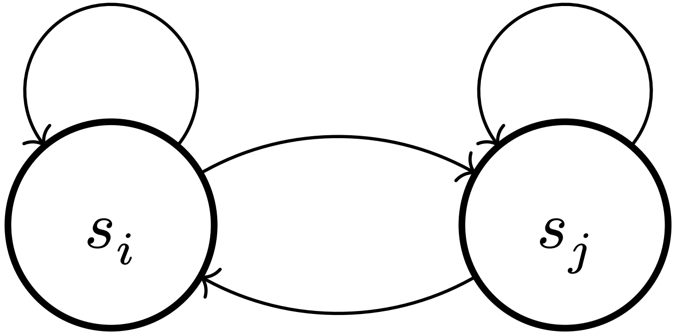
Mean reward over possible s_{n+1}

$$\mathbb{E}[R(s_{n+1}) \mid s_0, a_0, a_1, \dots a_n] = \sum_{s_1, \dots, s_{n+1} \in S} R(s_{n+1}) \prod_{t=0}^{n} \Pr(s_{t+1} \mid s_t, a_t)$$

Trajectory Optimization

$$
\begin{aligned}
\mathbb{E}[G \mid s_0, a_0, a_1, ...] = & \quad \mathbb{E}[R(s_1) \mid s_0, a_0] \\
+\gamma & \quad \mathbb{E}[R(s_2) \mid s_0, a_0, a_1] \\
+\gamma^2 & \quad \mathbb{E}[R(s_3) \mid s_0, a_0, a_1, a_2] \\
+ & \quad ... \\
= & \quad \sum_{s_1 \in S} R(s_1) \Pr(s_1 \mid s_0, a_0) \\
+\gamma & \quad \sum_{s_2 \in S} R(s_2) \sum_{s_1 \in S} \Pr(s_2 \mid s_1, a_1) \Pr(s_1 \mid s_0, a_0) \\
+\gamma^2 & \quad \sum_{s_3 \in S} R(s_3) \sum_{s_2 \in S} \Pr(s_3 \mid s_2, a_2) \sum_{s_1 \in S} \Pr(s_2 \mid s_1, a_1) ... \\
+ & ...
\end{aligned}
$$

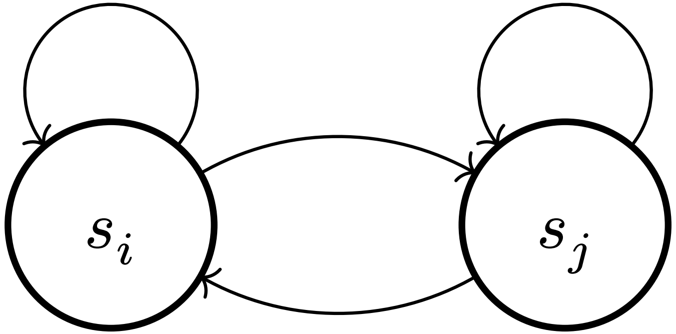
Trajectory Optimization



$R(s_i) = 0$ $R(s_j) = 1$

$$S = \{s_i, s_j\} \quad A = \{a_i, a_j\}$$

Trajectory Optimization

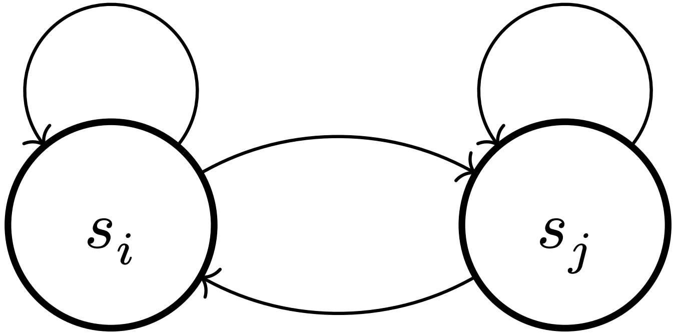


$R(s_i) = 0$ $\quad$ $R(s_j) = 1$

$$S = \{s_i, s_j\} \quad A = \{a_i, a_j\}$$

$$\Pr(s_i \mid s_i, a_i) = 0.8; \ \Pr(s_j \mid s_i, a_i) = 0.2$$

Trajectory Optimization



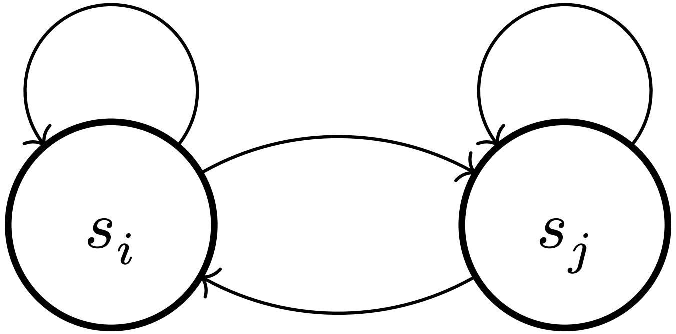
$R(s_i) = 0 \qquad R(s_j) = 1$

$$S = \{s_i, s_j\} \quad A = \{a_i, a_j\}$$

$$\Pr(s_i \mid s_i, a_i) = 0.8; \; \Pr(s_j \mid s_i, a_i) = 0.2$$

$$\Pr(s_i \mid s_i, a_j) = 0.7; \; \Pr(s_j \mid s_i, a_j) = 0.3$$

Trajectory Optimization



$R(s_i) = 0 \qquad R(s_j) = 1$

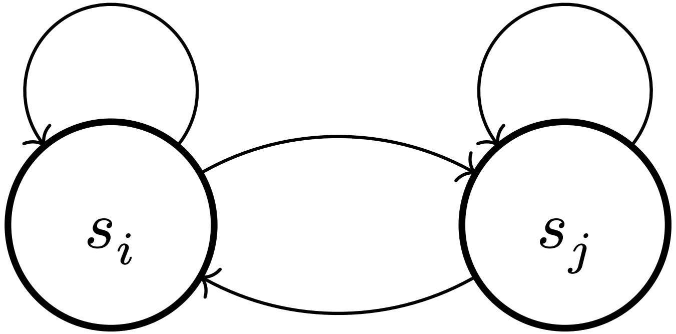
$$S = \{s_i, s_j\} \quad A = \{a_i, a_j\}$$

$$\Pr(s_i \mid s_i, a_i) = 0.8; \;\; \Pr(s_j \mid s_i, a_i) = 0.2$$

$$\Pr(s_i \mid s_i, a_j) = 0.7; \;\; \Pr(s_j \mid s_i, a_j) = 0.3$$

$$\Pr(s_i \mid s_j, a_i) = 0.6; \;\; \Pr(s_j \mid s_i, a_i) = 0.4$$

Trajectory Optimization



$R(s_i) = 0 \qquad R(s_j) = 1$

$$S = \{s_i, s_j\} \quad A = \{a_i, a_j\}$$

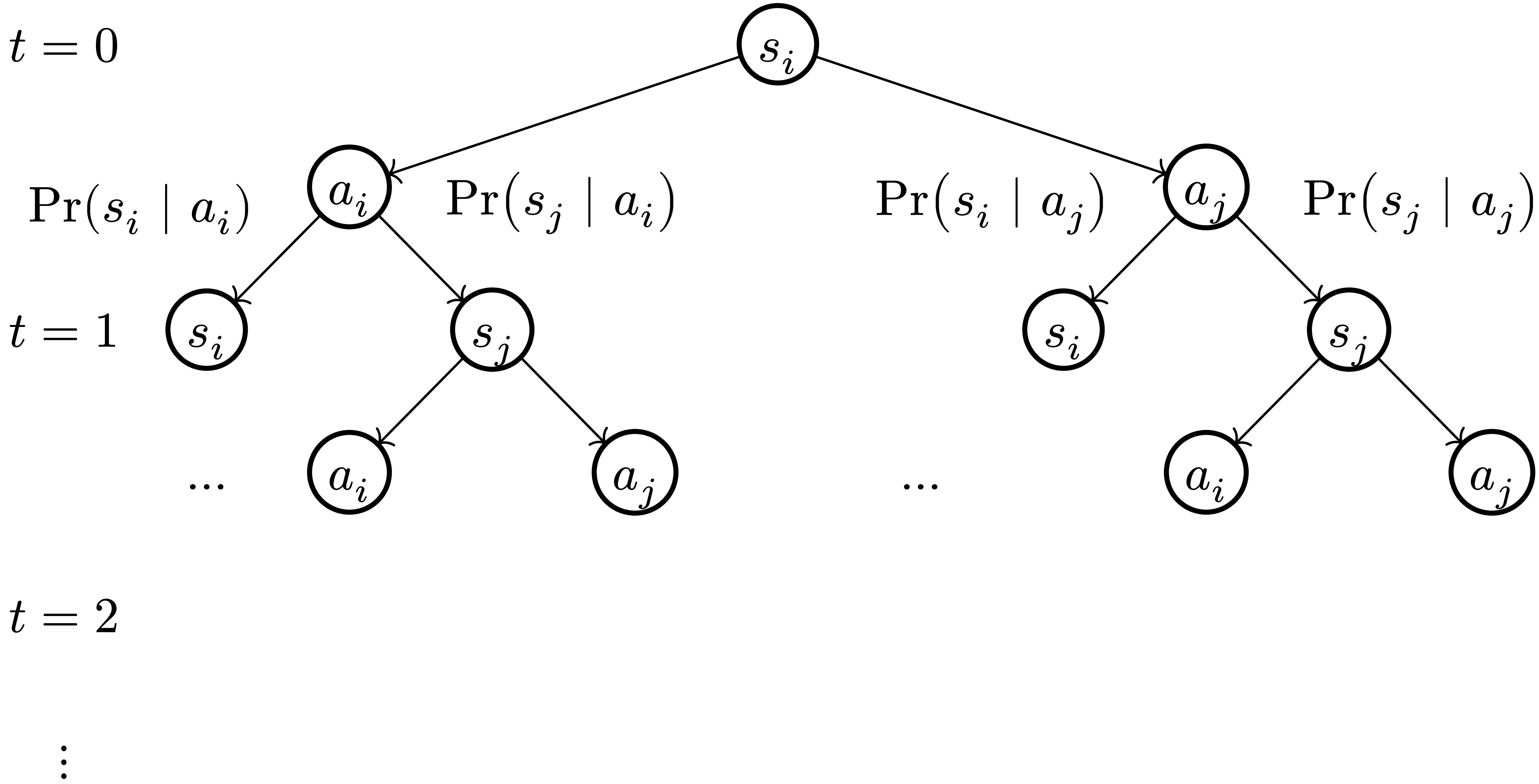
$$\Pr(s_i \mid s_i, a_i) = 0.8; \ \Pr(s_j \mid s_i, a_i) = 0.2$$

$$\Pr(s_i \mid s_i, a_j) = 0.7; \ \Pr(s_j \mid s_i, a_j) = 0.3$$

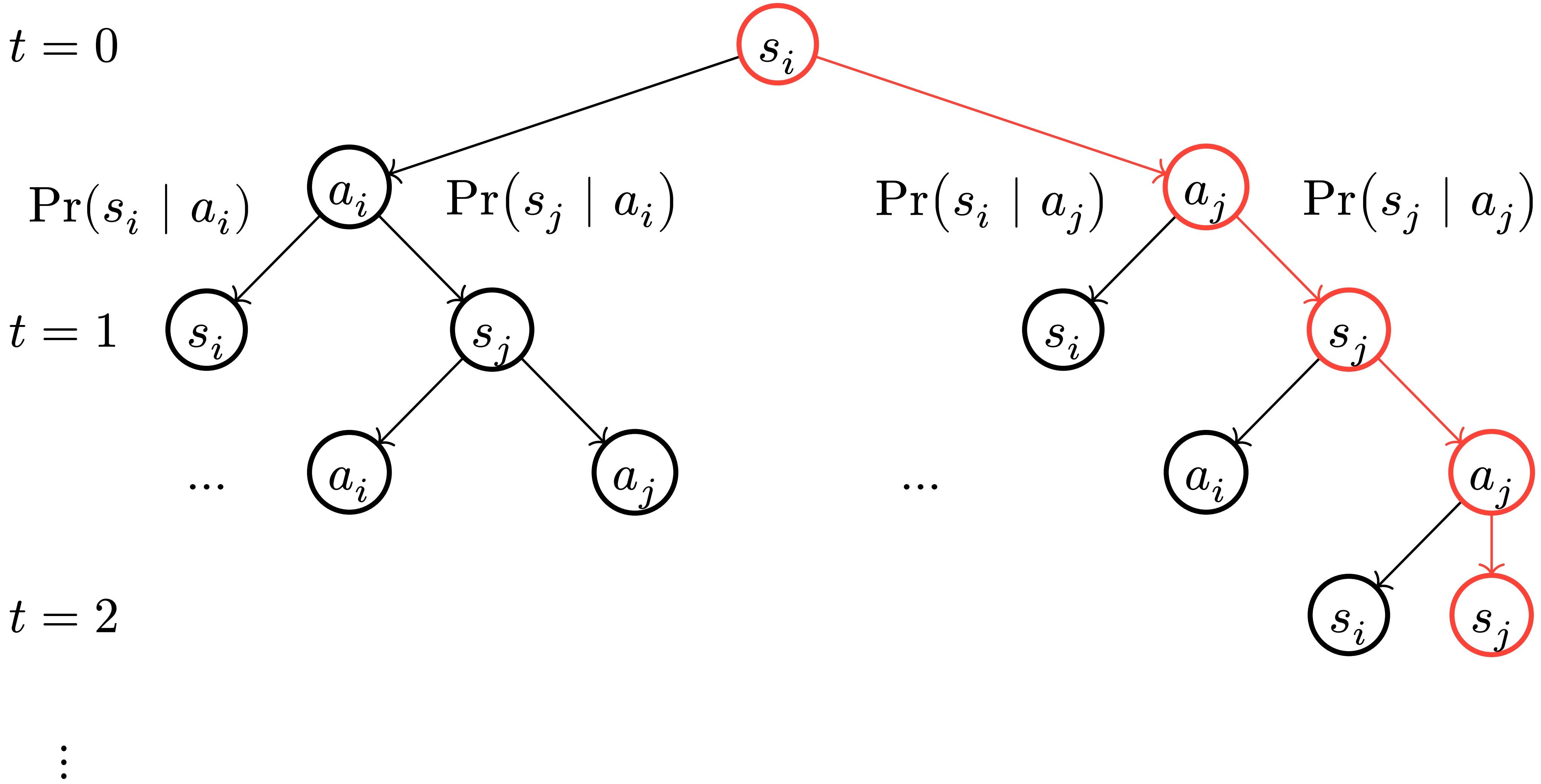
$$\Pr(s_i \mid s_j, a_i) = 0.6; \ \Pr(s_j \mid s_i, a_i) = 0.4$$

$$\Pr(s_i \mid s_j, a_j) = 0.1; \ \Pr(s_j \mid s_i, a_j) = 0.9$$

Trajectory Optimization



Trajectory Optimization



Trajectory Optimization

$$J(a_0, a_1, \ldots) = \mathbb{E}[G \mid s_0, a_0, a_1, \ldots] = \sum_{t=0}^{\infty} \gamma^t \sum_{s_{t+1} \in S} R(s_{t+1}) \cdot \Pr(s_{t+1} \mid s_t, a_t)$$

This expression gives us the **expected discounted return** J

Question: How can we maximize J?

$$\underset{a_0, a_1, \ldots \in A}{\arg\max}\, J(a_0, a_1, \ldots) = \underset{a_0, a_1, \ldots \in A}{\arg\max} \sum_{t=0}^{\infty} \gamma^t \sum_{s_{t+1} \in S} R(s_{t+1}) \cdot \Pr(s_{t+1} \mid s_t, a_t)$$

Trajectory Optimization

$$\operatorname*{arg\,max}_{a_0, a_1, \ldots \in A} J(a_0, a_1, \ldots) = \operatorname*{arg\,max}_{a_0, a_1, \ldots \in A} \sum_{t=0}^{\infty} \gamma^t \sum_{s_{t+1} \in S} R(s_{t+1}) \cdot \Pr(s_{t+1} \mid s_t, a_t)$$

In RL, we call this **trajectory optimization**

Question: What do we need to know about the problem to use trajectory optimization?

Answer:

- Must know the reward function R
- Must know the state transition function $T = \Pr(s_{t+1} \mid s_t, a_t)$

Trajectory Optimization

$$\operatorname*{arg\,max}_{a_0, a_1, \ldots \in A} J(a_0, a_1, \ldots) = \operatorname*{arg\,max}_{a_0, a_1, \ldots \in A} \sum_{t=0}^{\infty} \gamma^t \sum_{s_{t+1} \in S} R(s_{t+1}) \cdot \Pr(s_{t+1} \mid s_t, a_t)$$

Approach: Try all possible actions sequences and pick the one with the best return

Question: Any problem?

Answer: $a_0, a_1, \ldots$ is infinite, how can we try infinitely many actions?

We can't

Trajectory Optimization

$$\arg\max_{a_0, a_1, \ldots \in A} J(a_0, a_1, \ldots) = \arg\max_{a_0, a_1, \ldots \in A} \sum_{t=0}^{\infty} \gamma^t \sum_{s_{t+1} \in S} R(s_{t+1}) \cdot \Pr(s_{t+1} \mid s_t, a_t)$$

In trajectory optimization, we must introduce a **horizon** n

$$\arg\max_{a_0, a_1, \ldots, a_n \in A} J(a_0, a_1, \ldots, a_n) =$$

$$\arg\max_{a_0, a_1, \ldots a_n \in A} \sum_{t=0}^{n} \gamma^t \sum_{s_{t+1} \in S} R(s_{t+1}) \cdot \Pr(s_{t+1} \mid s_t, a_t)$$

Now, we can perform a search/optimization

Trajectory Optimization

$$\underset{a_0,...,a_n \in A}{\arg\max}\ J(a_0, ..., a_n) = \underset{a_0,...a_n, \in A}{\arg\max} \sum_{t=0}^{n} \gamma^t \sum_{s_{t+1} \in S} R(s_{t+1}) \cdot \Pr(s_{t+1} \mid s_t, a_t)$$

Question: What are the consequences of using a finite horizon n?

Answer:

- Our model can only consider rewards n steps into the future
- Actions will **not** be optimal

In certain cases, we do not care much about the distant future

Trajectory Optimization

$$\underset{a_0,...,a_n \in A}{\arg\max}\; J(a_0, ..., a_n) = \underset{a_0,...a_n, \in A}{\arg\max} \sum_{t=0}^{n} \gamma^t \sum_{s_{t+1} \in S} R(s_{t+1}) \cdot \Pr(s_{t+1} \mid s_t, a_t)$$

For example, we often use trajectory optimization to avoid crashes

If we can avoid any crash in 10 actions, then $n = 10$ is enough for us

One application of trajectory optimization:

https://www.youtube.com/watch?v=6qj3EfRTtkE

Trajectory Optimization

$$\underset{a_0, ..., a_n \in A}{\arg\max}\; J(a_0, ..., a_n) = \underset{a_0, ... a_n, \in A}{\arg\max} \sum_{t=0}^{n} \gamma^t \sum_{s_{t+1} \in S} R(s_{t+1}) \cdot \Pr(s_{t+1} \mid s_t, a_t)$$

How do we optimize J in practice?

- Try all possible sequences $a_0, ..., a_n$, pick the best one
- Randomly pick some sequences, pick the best one
- Use gradient descent to find $a_0, ..., a_n$
  - **Note:** The state transition function and reward function must be differentiable

Policies

Policies

With trajectory optimization, we plan all of our actions at once

$$\underset{a_0, a_1, \ldots \in A}{\arg\max}\ J(a_0, a_1, \ldots) = \underset{a_0, a_1, \ldots a_n \in A}{\arg\max} \sum_{t=0}^{n} \gamma^t \sum_{s_{t+1} \in S} R(s_{t+1}) \cdot \Pr(s_{t+1} \mid s_t, a_t)$$

It is difficult to think about many actions and states at once

Policies

To simplify, we introduce the **policy** π with parameters $\theta \in \Theta$

$$\pi : S \times \Theta \mapsto \Delta A$$

$$\Pr(a \mid s; \theta)$$

It maps a current state to a distribution of actions

The policy determines the behavior of our agent, it is the "brain"

Policies

$$J(a_0, a_1, \ldots) = \sum_{t=0}^{n} \gamma^t \sum_{s_{t+1} \in S} R(s_{t+1}) \cdot \Pr(s_{t+1} \mid s_t, a_t)$$

We can rewrite the expected return using the policy π and parameters θ

$$J(\theta) = \sum_{t=0}^{n} \gamma^t \sum_{s_{t+1} \in S} R(s_{t+1}) \cdot \Pr(s_{t+1} \mid s_t, a_t) \cdot \pi(a_t \mid s_t; \theta)$$

Policies

$$\underset{a_0, a_1, \ldots \in A}{\arg\max}\, J(a_0, a_1, \ldots) = \underset{a_0, a_1, \ldots a_n \in A}{\arg\max} \sum_{t=0}^{n} \gamma^t \sum_{s_{t+1} \in S} R(s_{t+1}) \cdot \Pr(s_{t+1} \mid s_t, a_t)$$

In controls and robotics, we call this **model-predictive control** (MPC)

Where do we use trajectory optimization/MPC?

https://www.youtube.com/watch?v=Kf9WDqYKYQQ

Policies

Trajectory optimization is expensive

The optimization process requires us to simulate thousands/millions of possible trajectories

However, as GPUs get faster these methods become more interesting

TODO: Visualization

TODO: What is the state transition function

Value Functions