Tutorial 1

Bayesian Optimization and Reinforcement Learning

Shadan Golestan
golestan@ualberta.ca

Machine Learning Paradigms

Machine Learning Paradigms

Supervised Learning

📊 **Data:** (x, y)

x: features, y: label

🎯 **Goal:** learn a function to map $y \rightarrow x$

💡 **Example:**

Cats: 😸😺😼

Dogs: 🐶🐶🐶

Machine Learning Paradigms

Supervised Learning

📊 **Data:** (x, y)

x: features, y: label

🎯 **Goal:** learn a function to map $y \rightarrow x$

💡 **Example:**

Cats: 😸😺😼

Dogs: 🐶🐶🐶

Unsupervised Learning

📊 **Data:** (x)

x: features, no label

🎯 **Goal:** learn underlying structure

💡 **Example:**

These are similar things 😸😺😼

These are similar things 🐶🐶🐶

Machine Learning Paradigms

Supervised Learning

📊 **Data:** (x, y)

x: features, y: label

🎯 **Goal:** learn a function to map $y \rightarrow x$

💡 **Example:**

Cats: 😸😺😼

Dogs: 🐶🐶🐶

Unsupervised Learning

📊 **Data:** (x)

x: features, no label

🎯 **Goal:** learn underlying structure

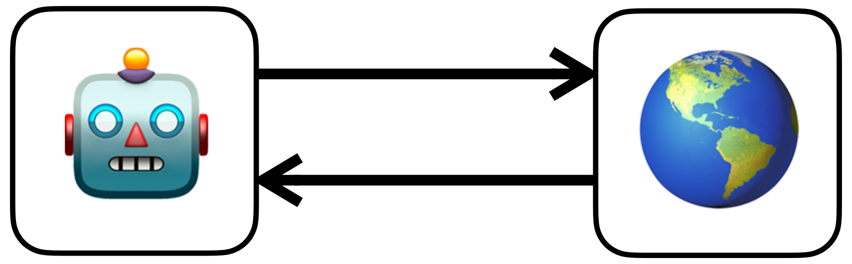
💡 **Example:**

These are similar things 😸😺😼

These are similar things 🐶🐶🐶

Sequential Decision Making

📊 **Data:**



🎯 **Goal:** maximize some reward

💡 **Example:**

🤖 eats 🍎 because it keeps 🤖 alive

Machine Learning Paradigms

Supervised Learning

📊 **Data:** (x, y)

x: features, y: label

🎯 **Goal:** learn a function to map $y \to x$

💡 **Example:**

Cats: 😺😺😼

Dogs: 🐶🐶🐶

Unsupervised Learning

📊 **Data:** (x)

x: features, no label

🎯 **Goal:** learn underlying structure

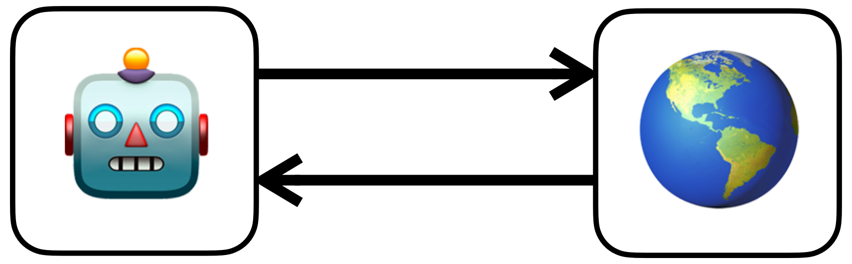
💡 **Example:**

These are similar things 😺😺😼

These are similar things 🐶🐶🐶

Sequential Decision Making

📊 **Data:**

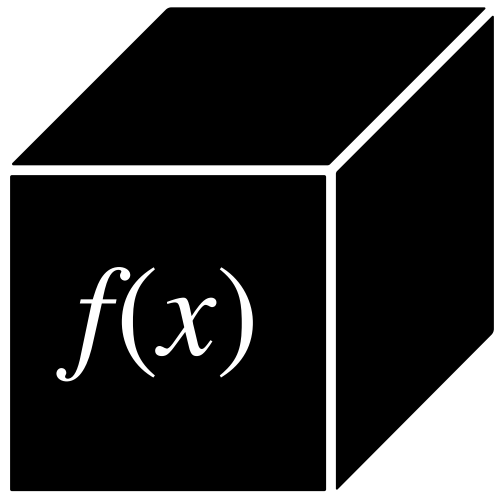


🎯 **Goal:** maximize some reward

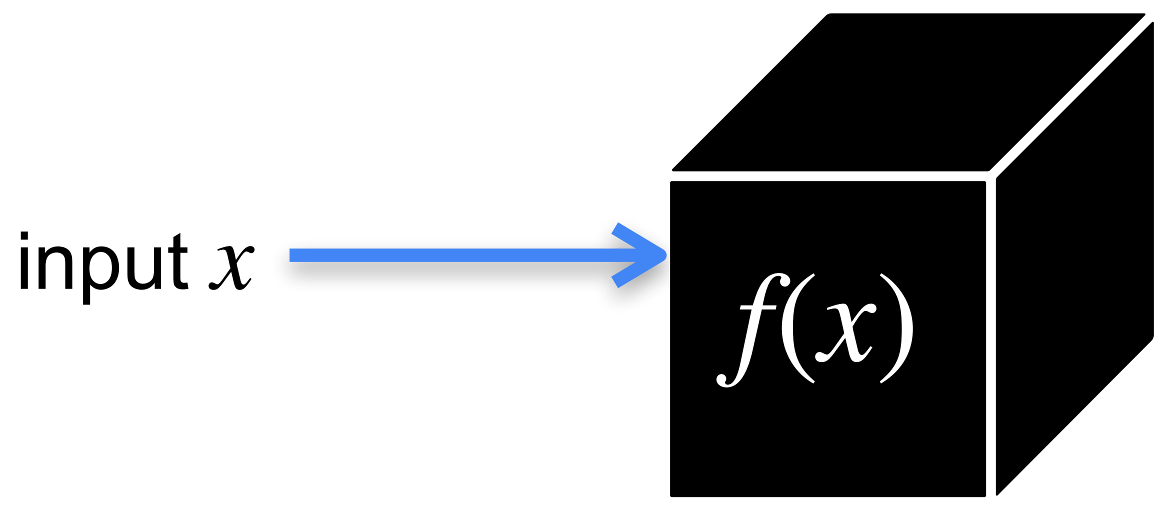
💡 **Example:**

🤖 eats 🍎 because it keeps 🤖 alive

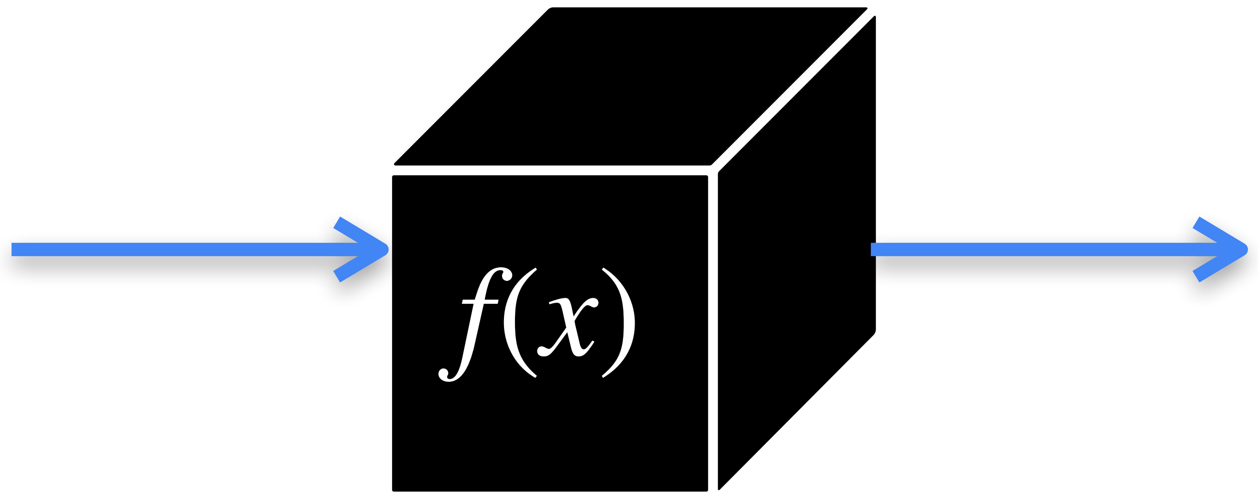
Sequential Decision Making



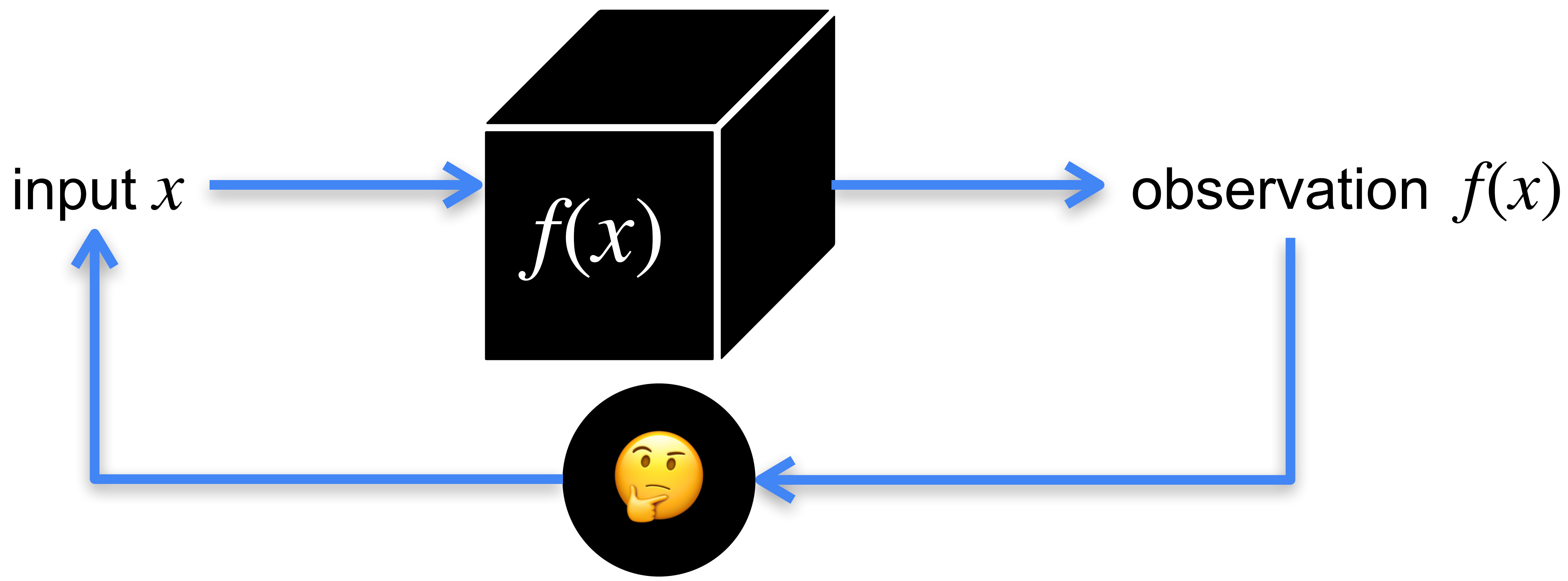
Sequential Decision Making



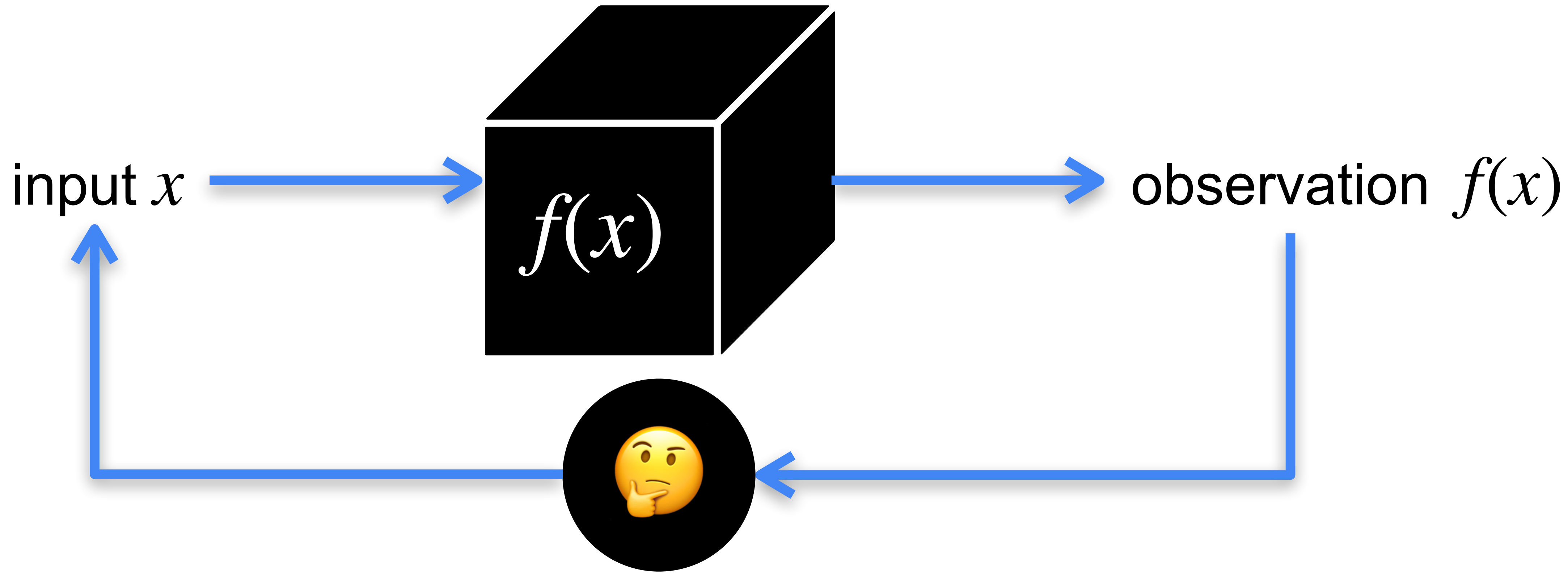
Sequential Decision Making

input x  observation $f(x)$

Sequential Decision Making



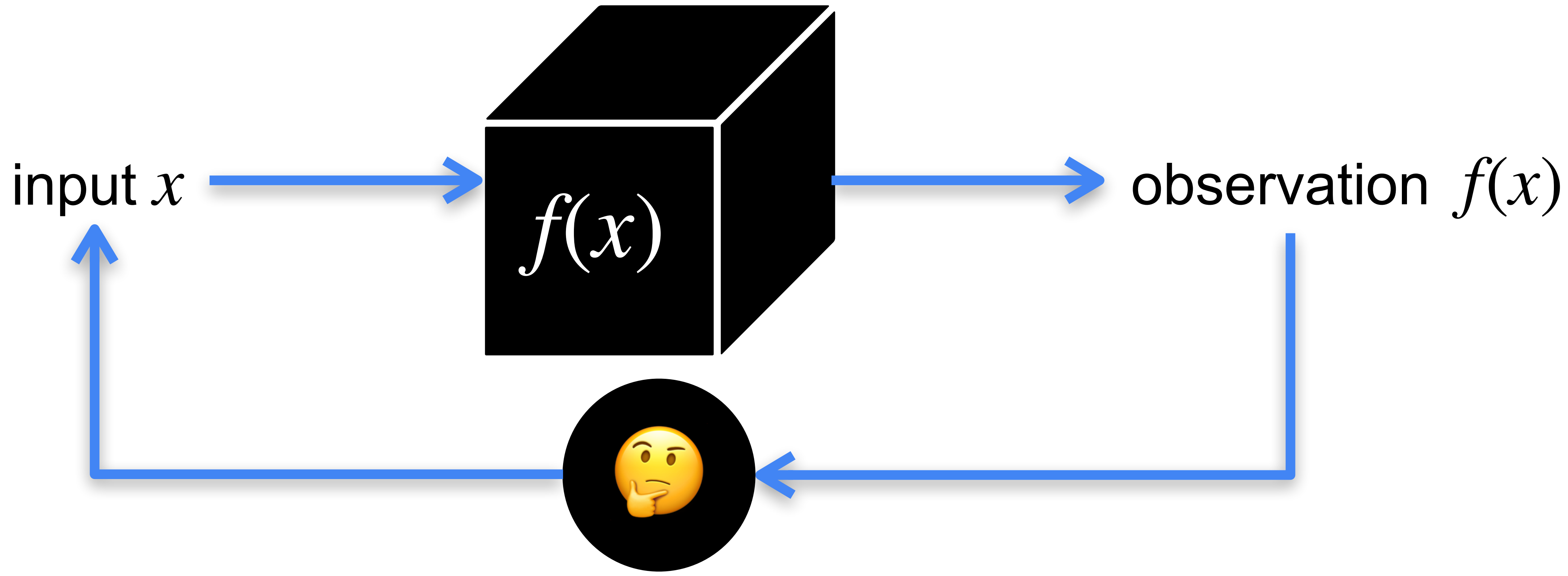
Sequential Decision Making



Estimation of Distribution Algorithms (Bayesian Optimization)

Reinforcement Learning

Sequential Decision Making



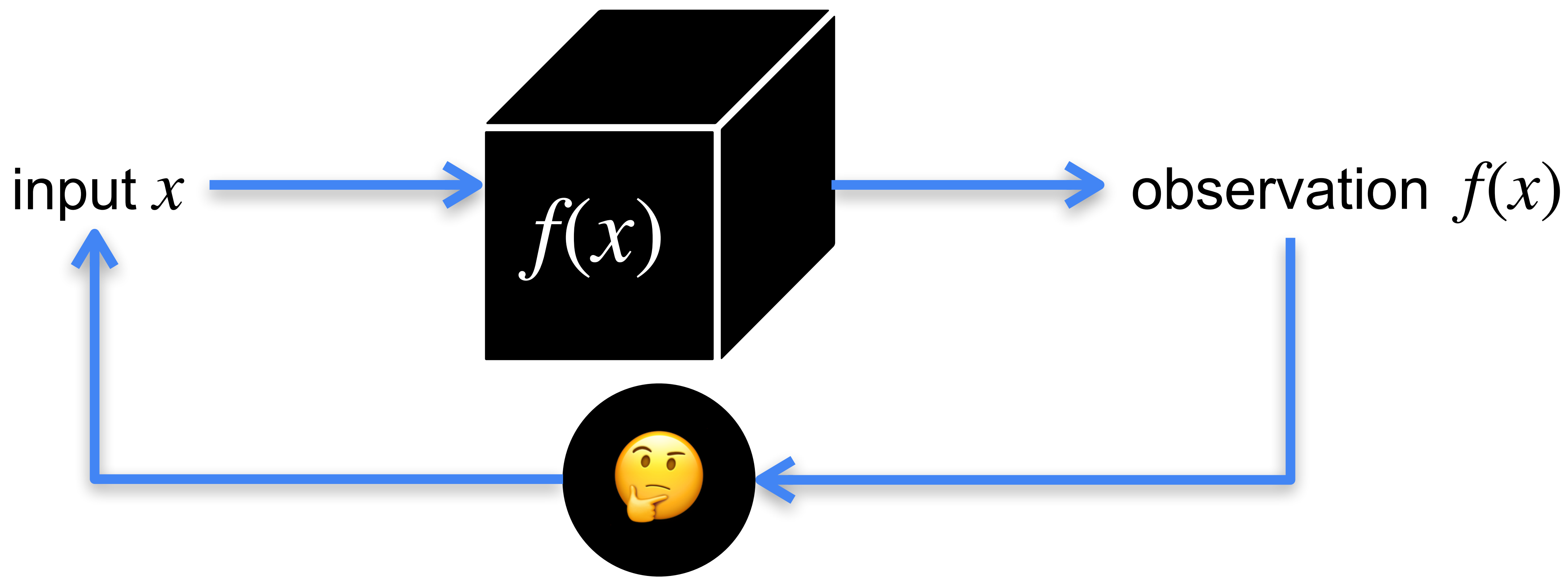
Estimation of Distribution Algorithms (Bayesian Optimization)

📦 **Function** $f(x)$

does not change with x (static)

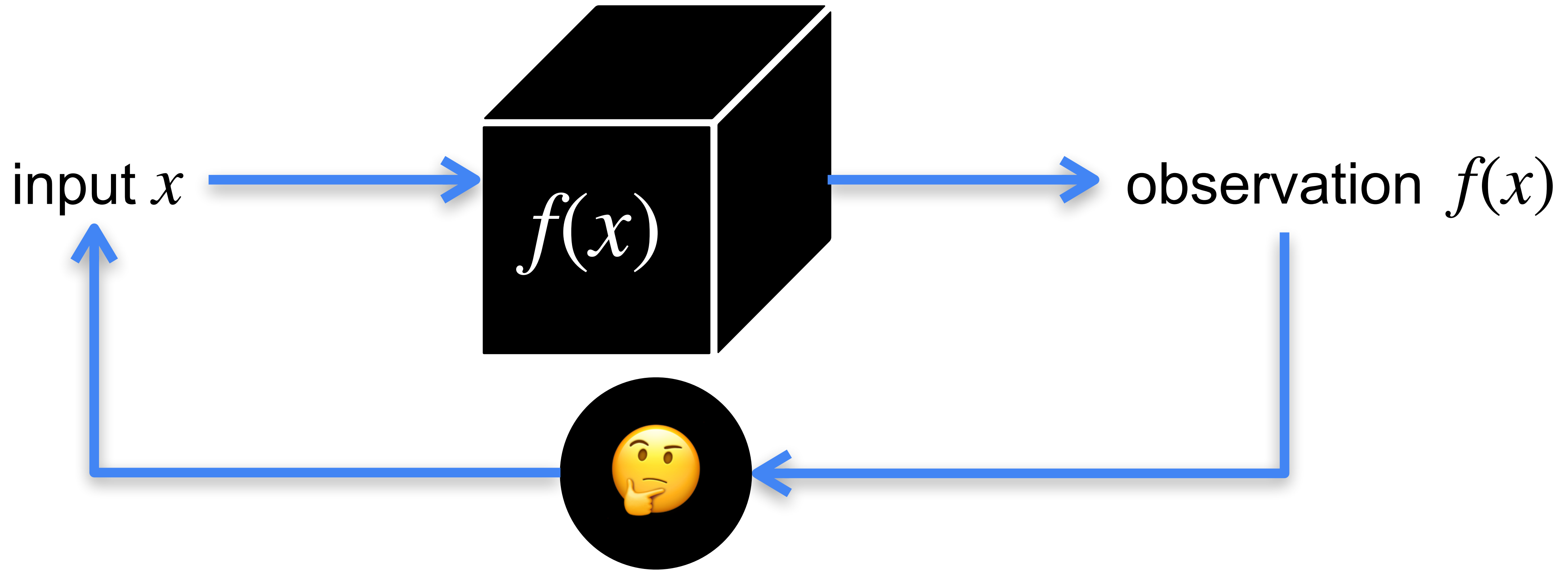
Reinforcement Learning

Sequential Decision Making



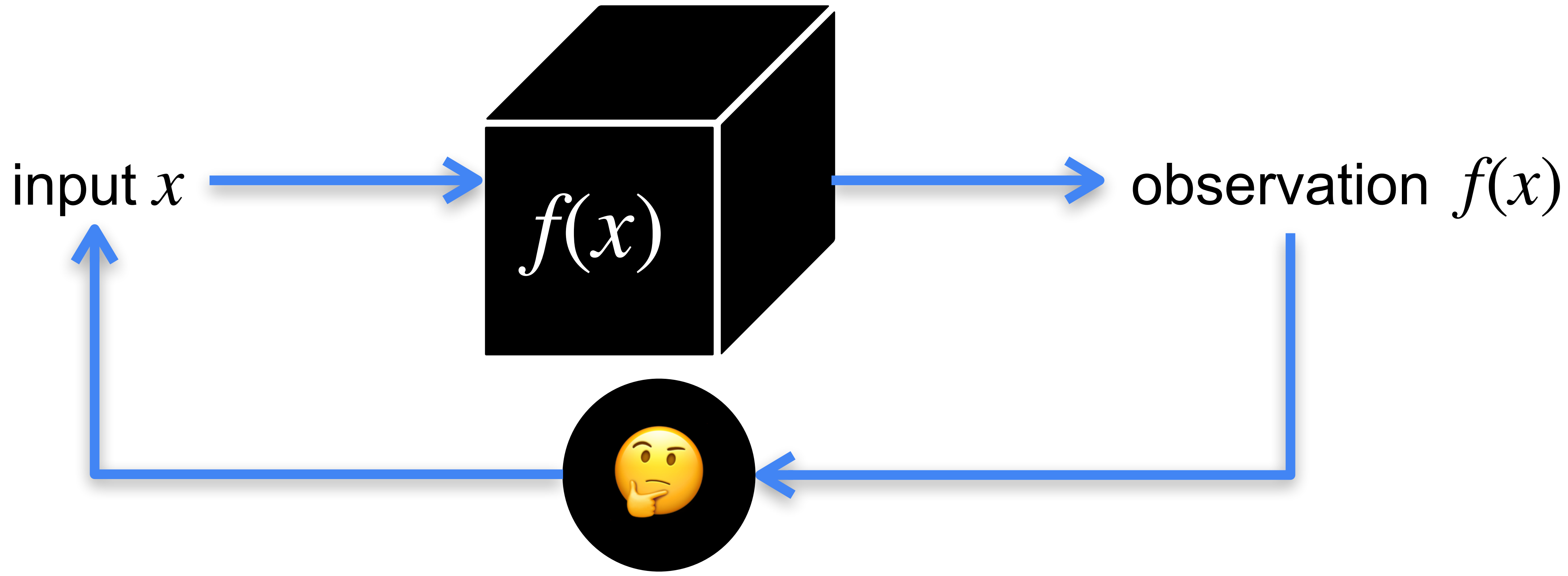
Estimation of Distribution Algorithms (Bayesian Optimization)	Reinforcement Learning
📦 **Function** $f(x)$ does not change with x (static)	📦 **Function** $f(x)$ changes with x (dynamic)

Sequential Decision Making



Estimation of Distribution Algorithms (Bayesian Optimization)	Reinforcement Learning
📦 **Function** $f(x)$ does not change with x (static)	📦 **Function** $f(x)$ changes with x (dynamic)
🎯 **Goal:** finding the best x	

Sequential Decision Making



Estimation of Distribution Algorithms (Bayesian Optimization)

📦 **Function** $f(x)$ does not change with x (static)

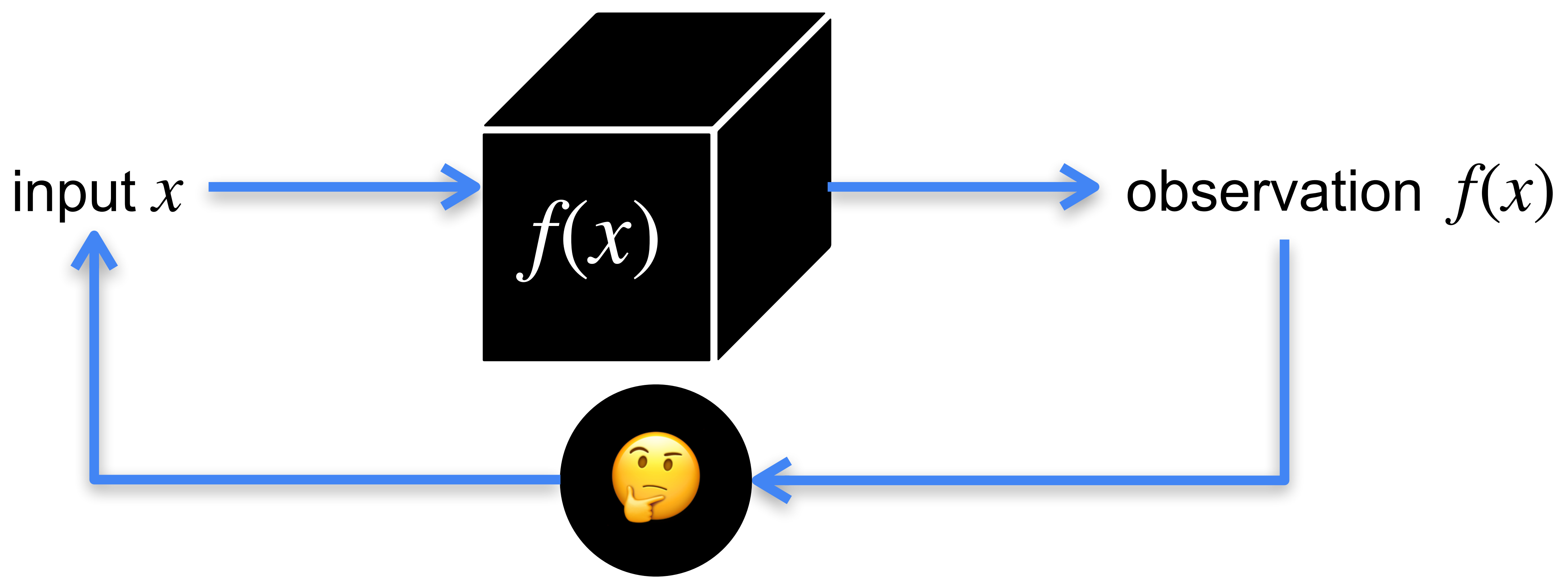
🎯 **Goal:** finding the best x

Reinforcement Learning

📦 **Function** $f(x)$ changes with x (dynamic)

🎯 **Goal:** finding a policy that gives the best x at each state of $f(x)$

Sequential Decision Making



Estimation of Distribution Algorithms (Bayesian Optimization)

📦 **Function** $f(x)$
does not change with x (static)

🎯 **Goal:** finding the best x

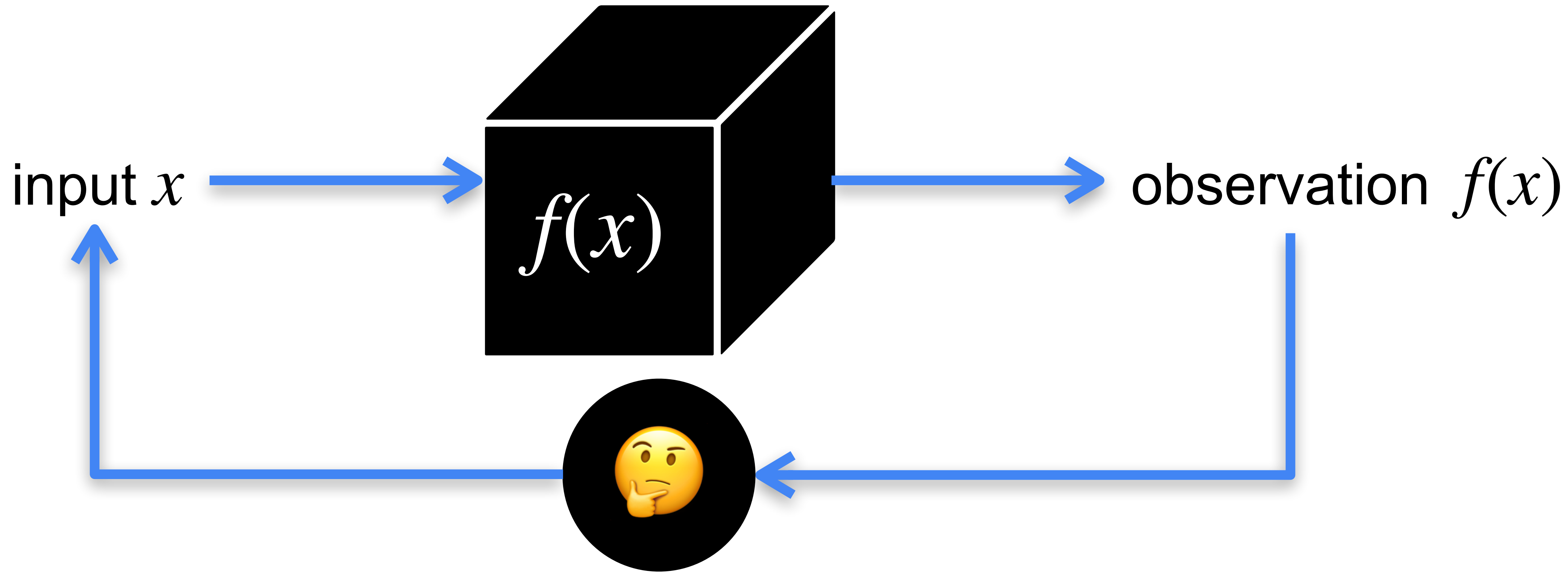
⚙️ **Process:** learn and improve decisions iteratively

Reinforcement Learning

📦 **Function** $f(x)$ changes with x (dynamic)

🎯 **Goal:** finding a policy that gives the best x at each state of $f(x)$

Sequential Decision Making



Estimation of Distribution Algorithms (Bayesian Optimization)

📦 **Function** $f(x)$
does not change with x (static)

🎯 **Goal:** finding the best x

⚙️ **Process:** learn and improve decisions iteratively

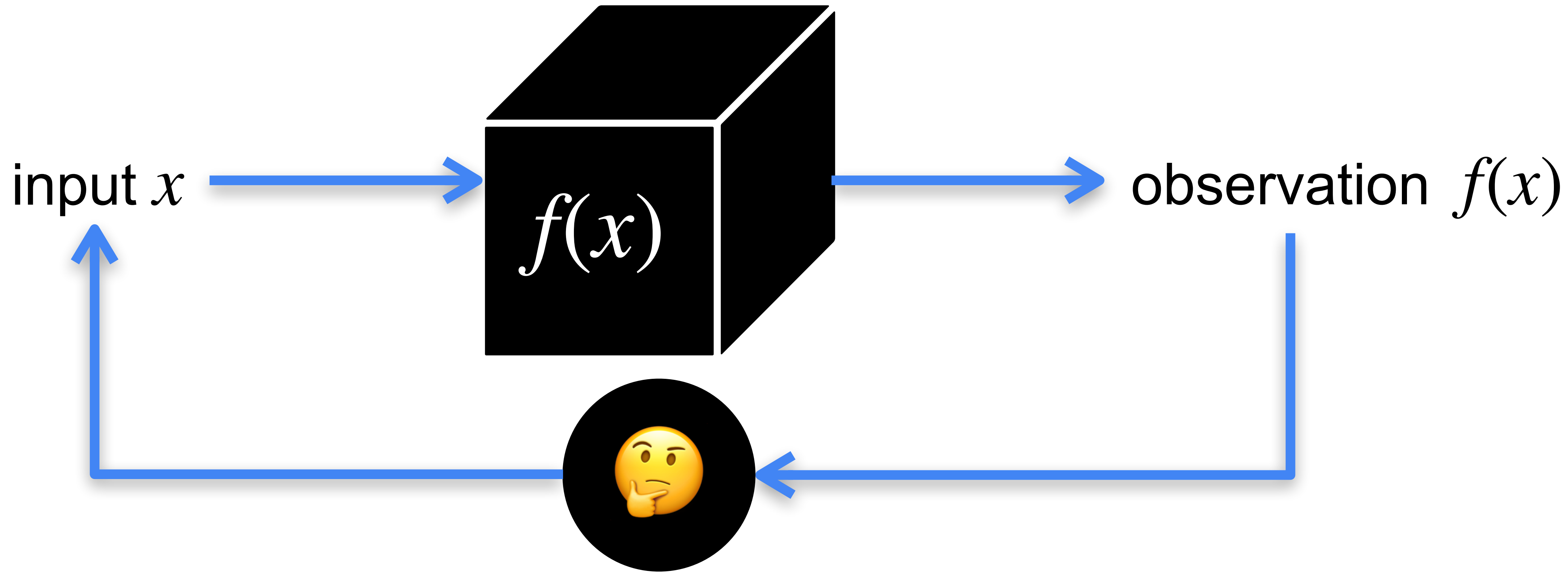
Reinforcement Learning

📦 **Function** $f(x)$ changes with x (dynamic)

🎯 **Goal:** finding a policy that gives the best x at each state of $f(x)$

⚙️ **Process:** learn and improve decisions iteratively

Sequential Decision Making



Estimation of Distribution Algorithms (Bayesian Optimization)

📦 **Function** $f(x)$ does not change with x (static)

🎯 **Goal:** finding the best x

⚙️ **Process:** learn and improve decisions iteratively

🤷 **Uncertainty:** unknown, noisy $f(x)$

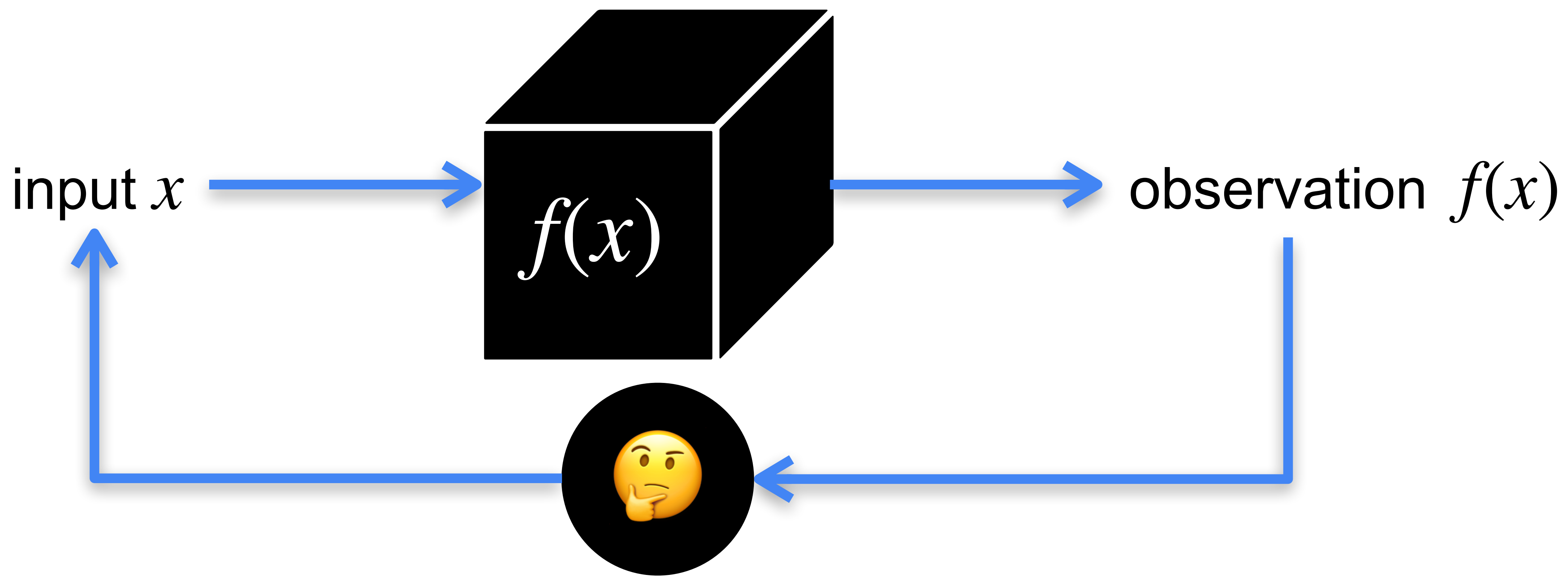
Reinforcement Learning

📦 **Function** $f(x)$ changes with x (dynamic)

🎯 **Goal:** finding a policy that gives the best x at each state of $f(x)$

⚙️ **Process:** learn and improve decisions iteratively

Sequential Decision Making



Estimation of Distribution Algorithms (Bayesian Optimization)

📦 **Function** $f(x)$ does not change with x (static)

🎯 **Goal:** finding the best x

⚙️ **Process:** learn and improve decisions iteratively

🤷‍♂️ **Uncertainty:** unknown, noisy $f(x)$

Reinforcement Learning

📦 **Function** $f(x)$ changes with x (dynamic)

🎯 **Goal:** finding a policy that gives the best x at each state of $f(x)$

⚙️ **Process:** learn and improve decisions iteratively

🤷‍♂️ **Uncertainty:** unknown, noisy $f(x)$

Estimation of Distribution Algorithms (Bayesian Optimization)

Function $f(x)$ does not change with x (static)

Goal: finding the best x

Process: learn and improve decisions iteratively

Uncertainty: unknown, noisy $f(x)$

Reinforcement Learning

Function $f(x)$ changes with x (dynamic)

Goal: finding a policy that gives the best x at each state of $f(x)$

Process: learn and improve decisions iteratively

Uncertainty: unknown, noisy $f(x)$

Estimation of Distribution Algorithms (Bayesian Optimization)

📦 **Function** $f(x)$ does not change with x (static)

🎯 **Goal:** finding the best x

⚙️ **Process:** learn and improve decisions iteratively

🤷 **Uncertainty:** unknown, noisy $f(x)$

🎰 🎰 🎰 🎰

Reinforcement Learning

📦 **Function** $f(x)$ changes with x (dynamic)

🎯 **Goal:** finding a policy that gives the best x at each state of $f(x)$

⚙️ **Process:** learn and improve decisions iteratively

🤷 **Uncertainty:** unknown, noisy $f(x)$

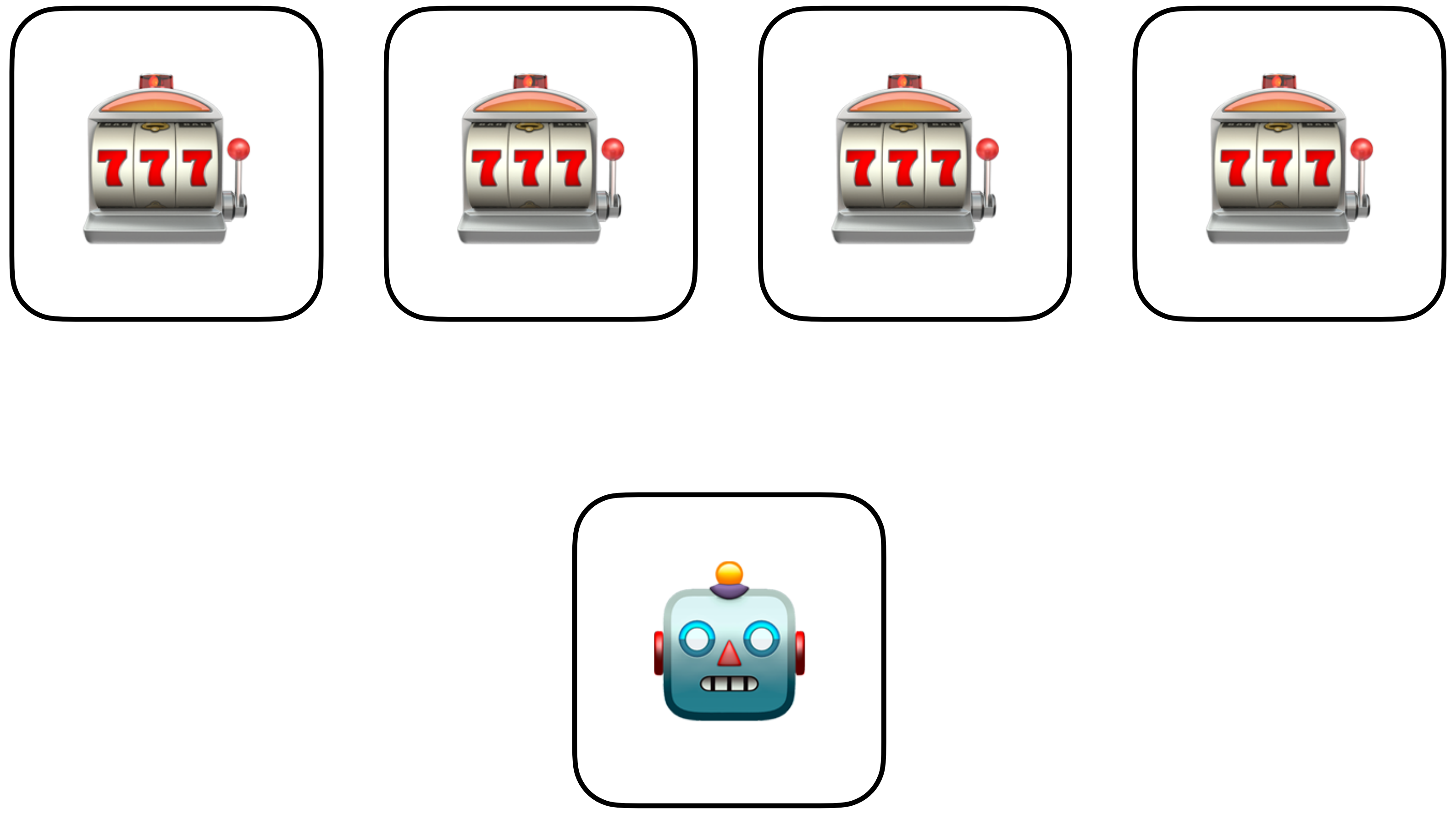
Estimation of Distribution Algorithms (Bayesian Optimization)

Function $f(x)$ does not change with x (static)

Goal: finding the best x

Process: learn and improve decisions iteratively

Uncertainty: unknown, noisy $f(x)$



Reinforcement Learning

Function $f(x)$ changes with x (dynamic)

Goal: finding a policy that gives the best x at each state of $f(x)$

Process: learn and improve decisions iteratively

Uncertainty: unknown, noisy $f(x)$

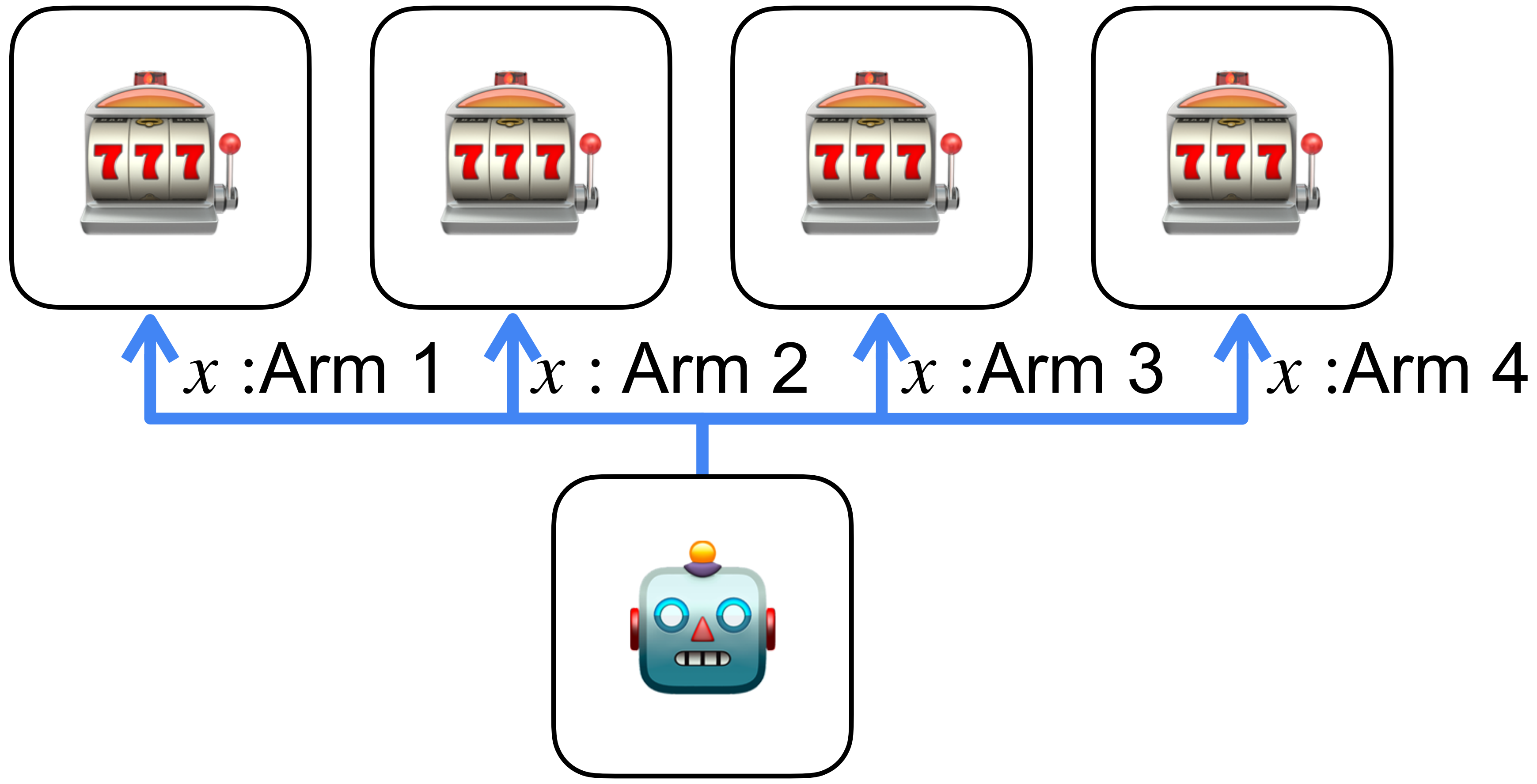
Estimation of Distribution Algorithms (Bayesian Optimization)

📦 **Function** $f(x)$ does not change with x (static)

🎯 **Goal:** finding the best x

⚙️ **Process:** learn and improve decisions iteratively

🤷‍♂️ **Uncertainty:** unknown, noisy $f(x)$



Reinforcement Learning

📦 **Function** $f(x)$ changes with x (dynamic)

🎯 **Goal:** finding a policy that gives the best x at each state of $f(x)$

⚙️ **Process:** learn and improve decisions iteratively

🤷‍♂️ **Uncertainty:** unknown, noisy $f(x)$

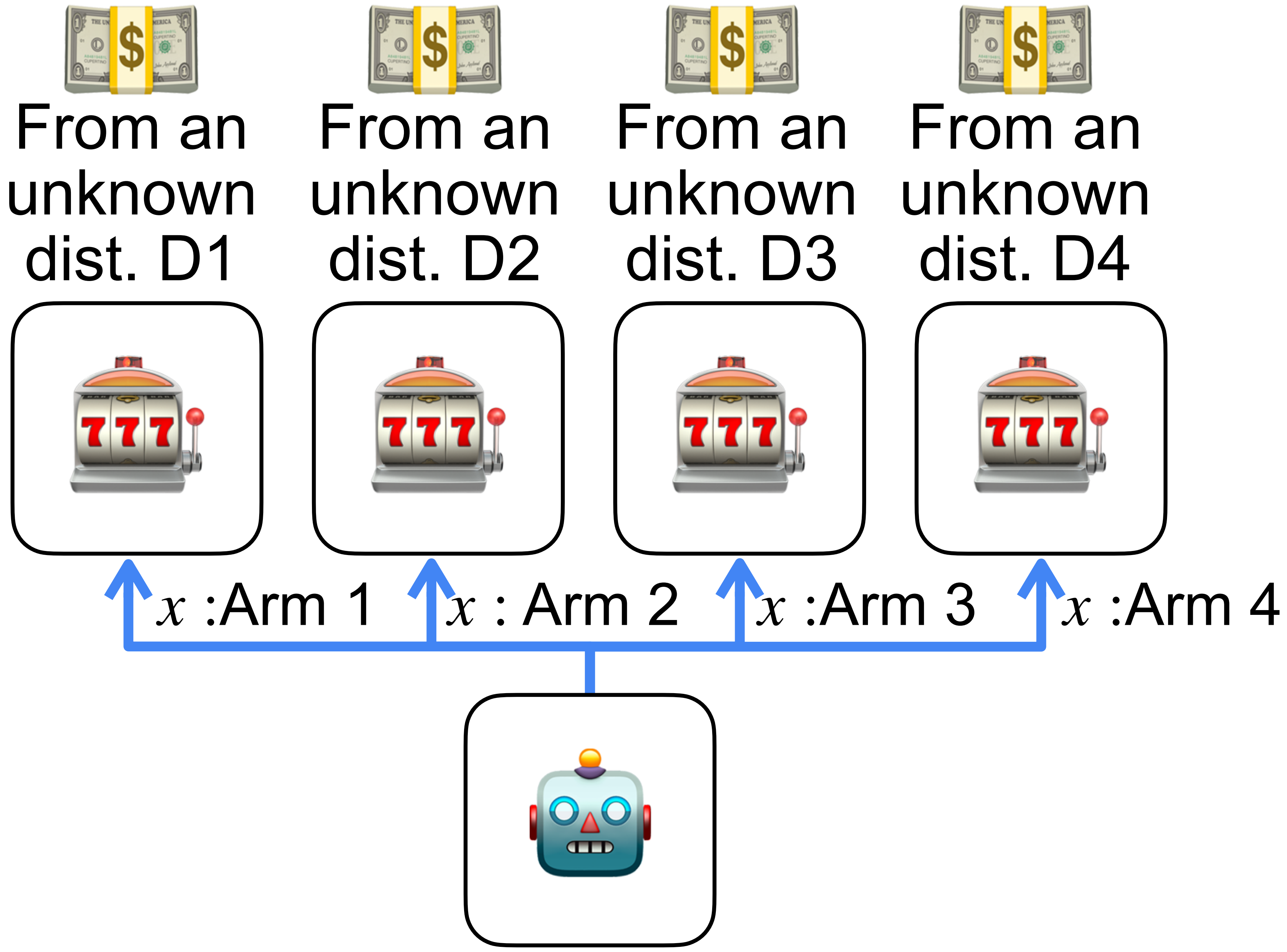
Estimation of Distribution Algorithms (Bayesian Optimization)

📦 **Function** $f(x)$ does not change with x (static)

🎯 **Goal:** finding the best x

⚙️ **Process:** learn and improve decisions iteratively

🤷 **Uncertainty:** unknown, noisy $f(x)$



Reinforcement Learning

📦 **Function** $f(x)$ changes with x (dynamic)

🎯 **Goal:** finding a policy that gives the best x at each state of $f(x)$

⚙️ **Process:** learn and improve decisions iteratively

🤷 **Uncertainty:** unknown, noisy $f(x)$

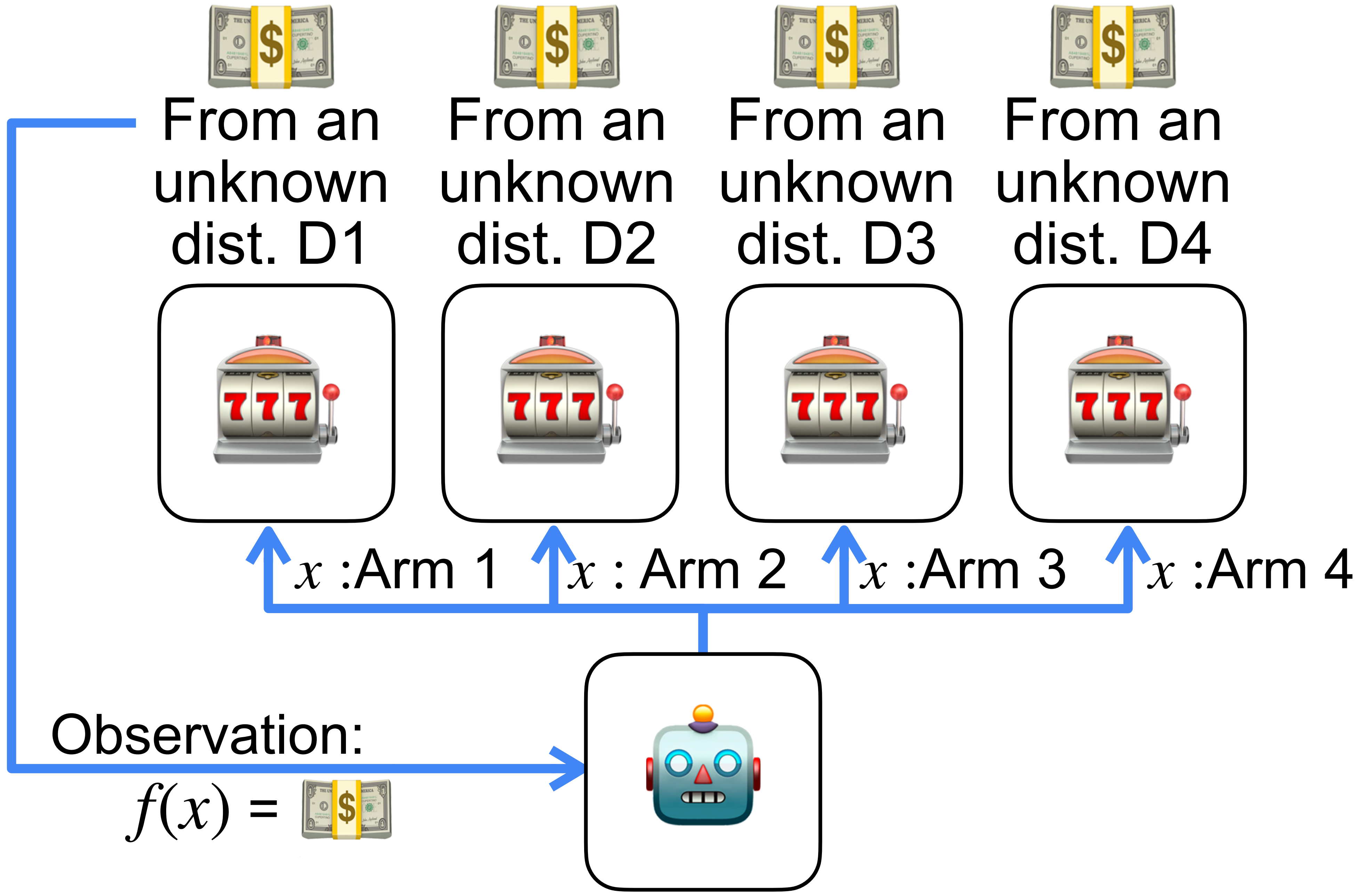
Estimation of Distribution Algorithms (Bayesian Optimization)

📦 **Function** $f(x)$ does not change with x (static)

🎯 **Goal:** finding the best x

⚙️ **Process:** learn and improve decisions iteratively

🤷‍♂️ **Uncertainty:** unknown, noisy $f(x)$



Reinforcement Learning

📦 **Function** $f(x)$ changes with x (dynamic)

🎯 **Goal:** finding a policy that gives the best x at each state of $f(x)$

⚙️ **Process:** learn and improve decisions iteratively

🤷‍♂️ **Uncertainty:** unknown, noisy $f(x)$

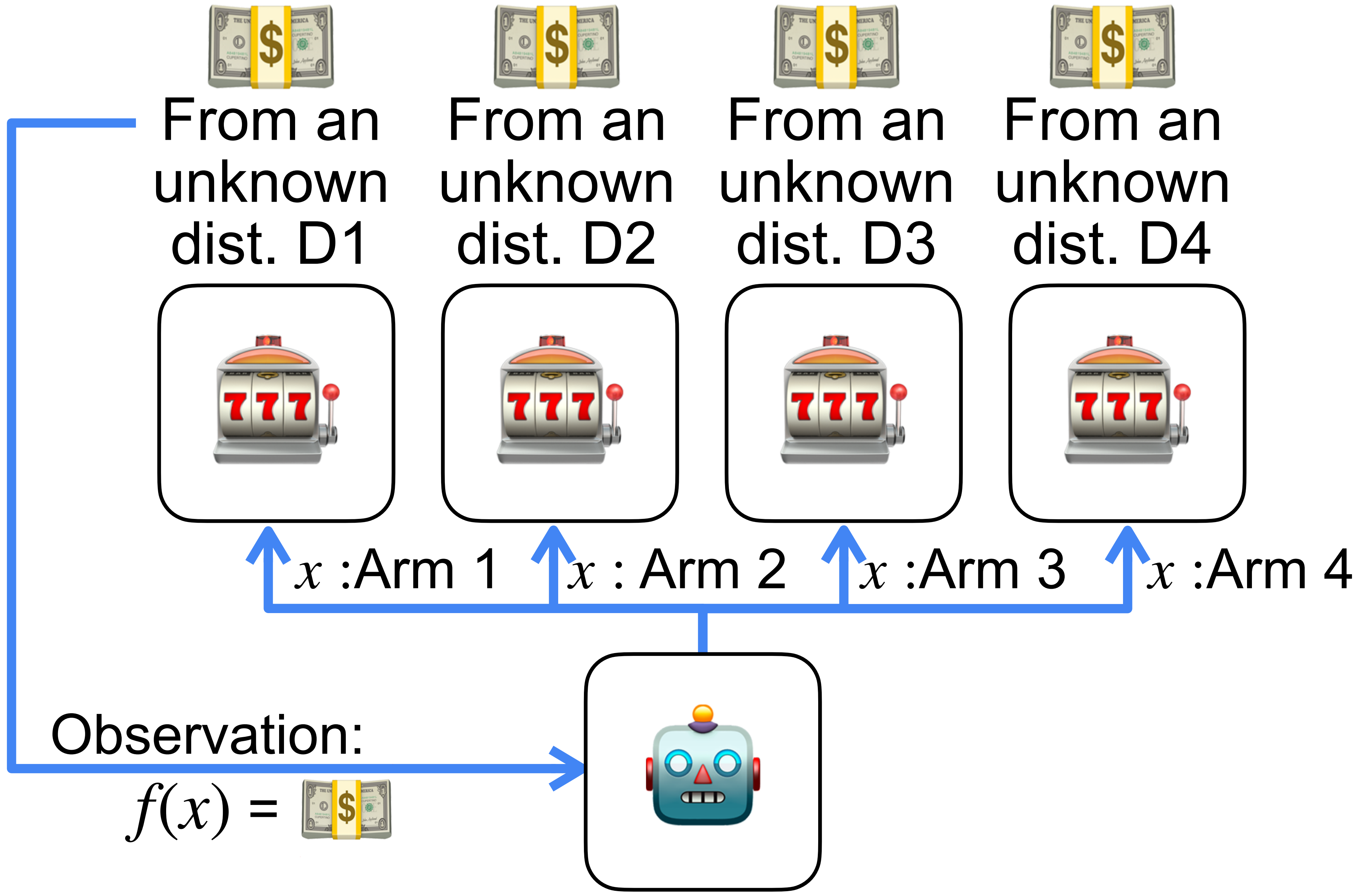
Estimation of Distribution Algorithms (Bayesian Optimization)

Function $f(x)$ does not change with x (static)

Goal: finding the best x

Process: learn and improve decisions iteratively

Uncertainty: unknown, noisy $f(x)$



Reinforcement Learning

Function $f(x)$ changes with x (dynamic)

Goal: finding a policy that gives the best x at each state of $f(x)$

Process: learn and improve decisions iteratively

Uncertainty: unknown, noisy $f(x)$

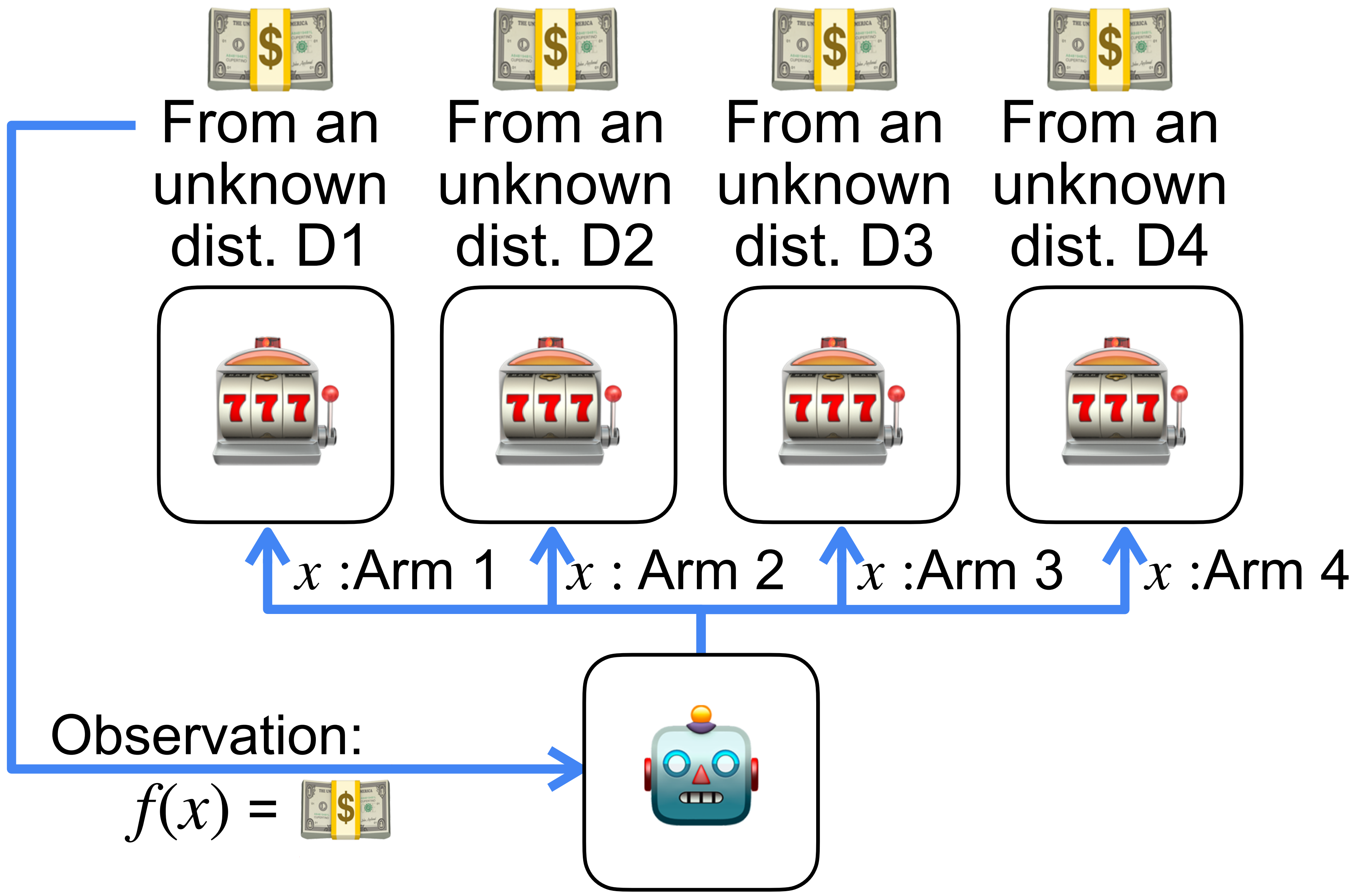
Estimation of Distribution Algorithms (Bayesian Optimization)

📦 **Function** $f(x)$ does not change with x (static)

🎯 **Goal:** finding the best x

⚙️ **Process:** learn and improve decisions iteratively

🤷‍♂️ **Uncertainty:** unknown, noisy $f(x)$



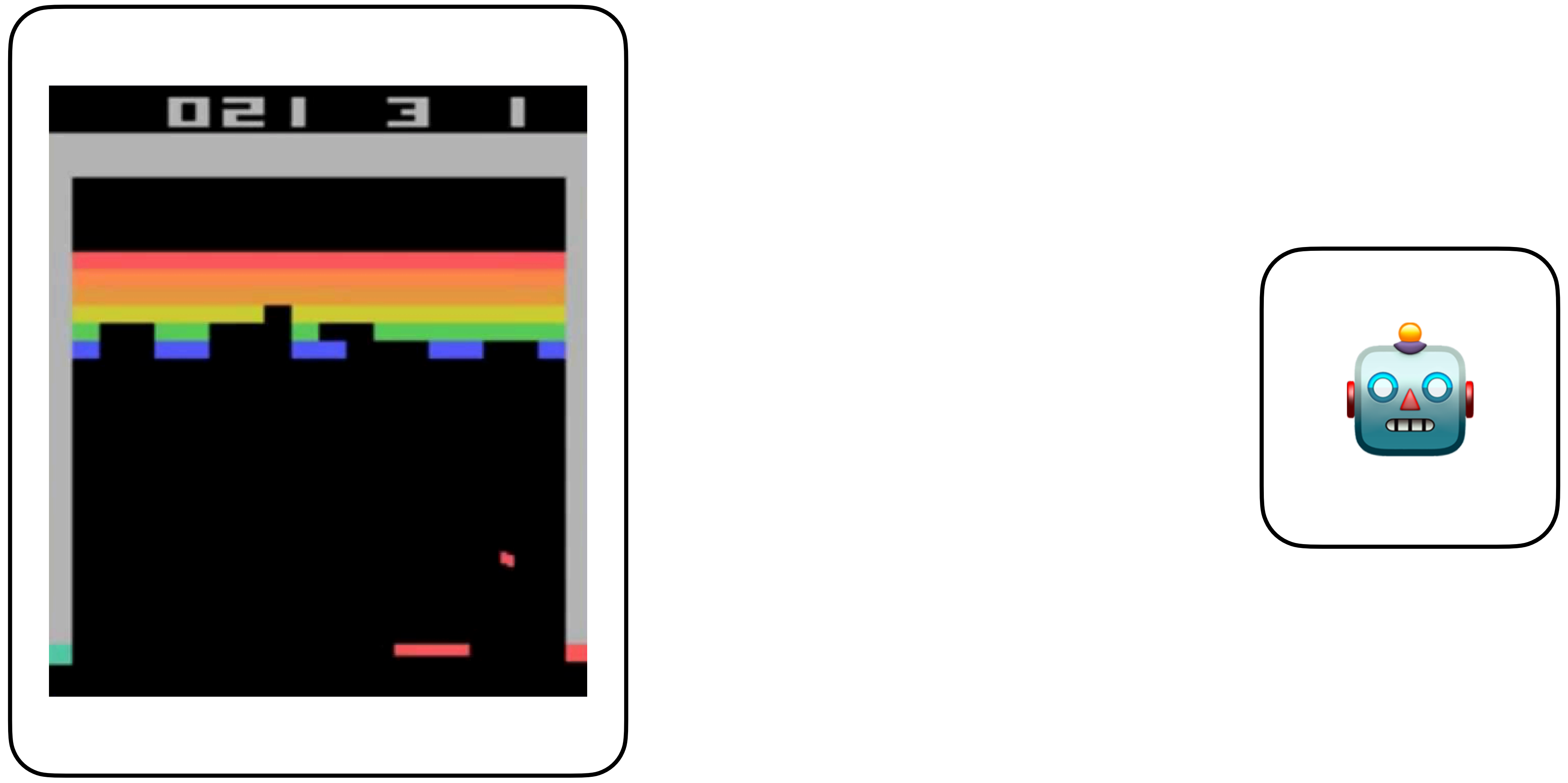
Reinforcement Learning

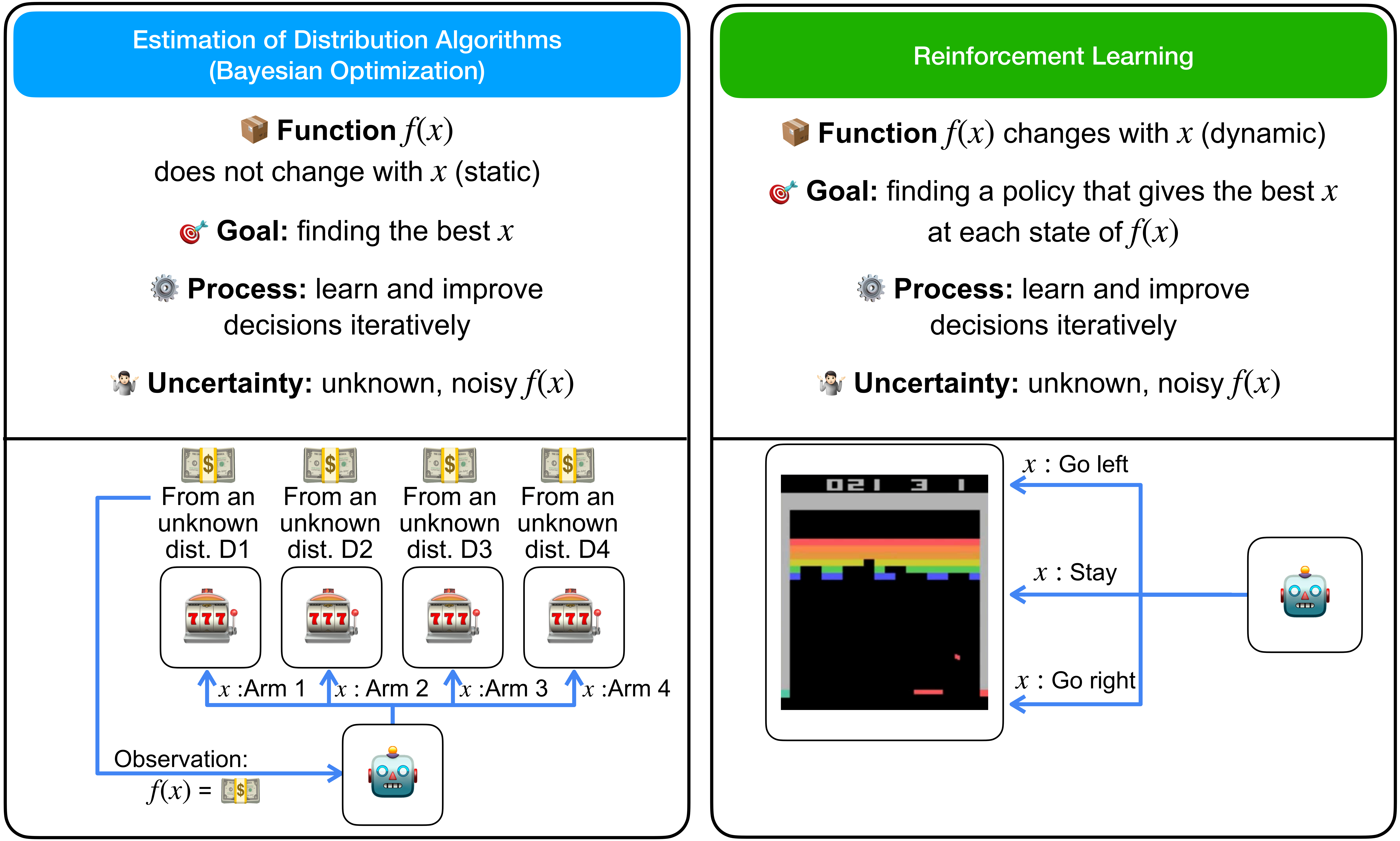
📦 **Function** $f(x)$ changes with x (dynamic)

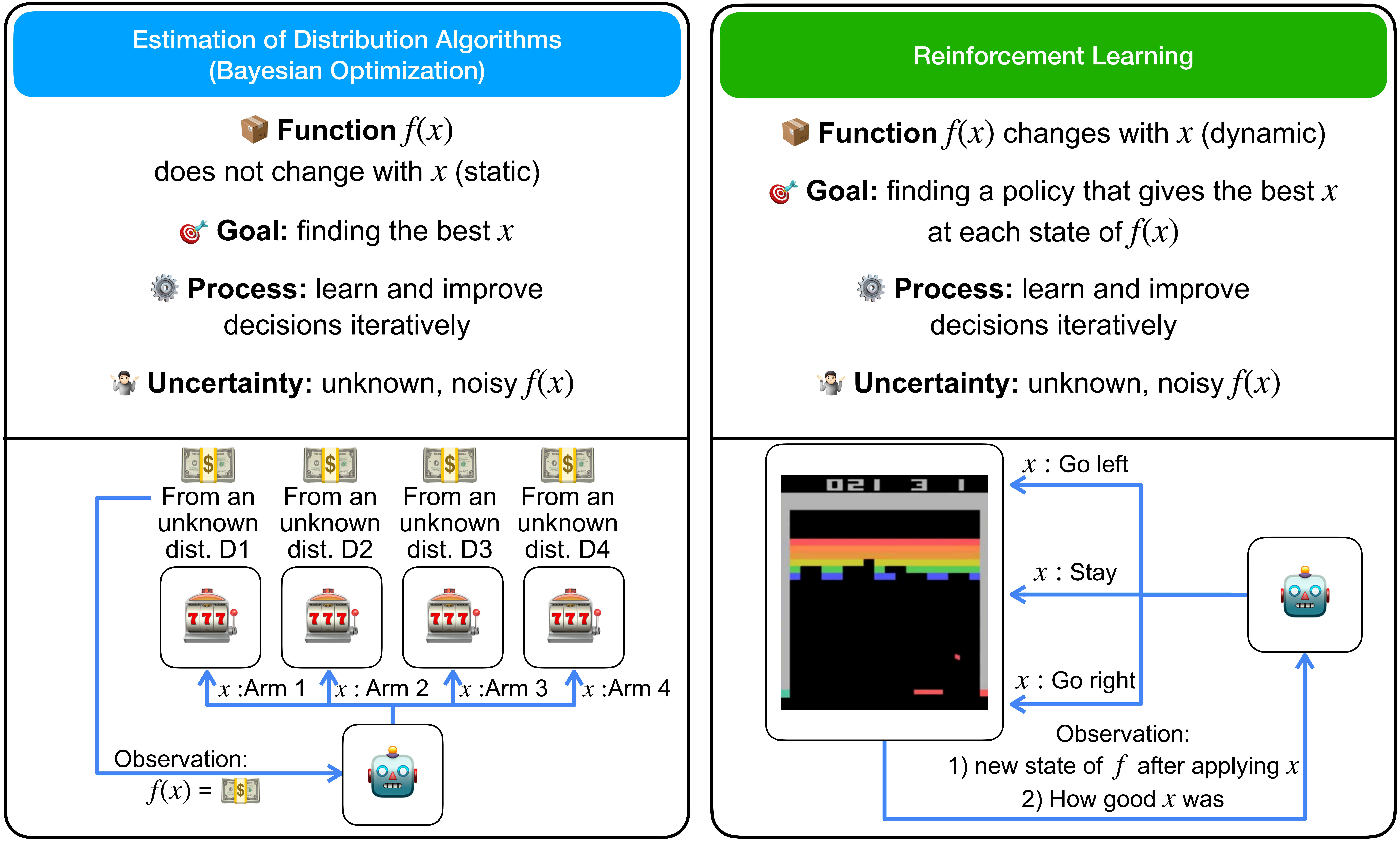
🎯 **Goal:** finding a policy that gives the best x at each state of $f(x)$

⚙️ **Process:** learn and improve decisions iteratively

🤷‍♂️ **Uncertainty:** unknown, noisy $f(x)$




Estimation of Distribution Algorithms
(Bayesian Optimization)
Function $f(x)$
does not change with x (static)
Goal: finding the best x
Process: learn and improve
decisions iteratively
Uncertainty: unknown, noisy $f(x)$
From an unknown dist. D1
From an unknown dist. D2
From an unknown dist. D3
From an unknown dist. D4
x :Arm 1
x : Arm 2
x :Arm 3
x :Arm 4
Observation:
$f(x)$ =
Reinforcement Learning
Function $f(x)$ changes with x (dynamic)
Goal: finding a policy that gives the best x
at each state of $f(x)$
Process: learn and improve
decisions iteratively
Uncertainty: unknown, noisy $f(x)$
x : Go left
x : Stay
x : Go right

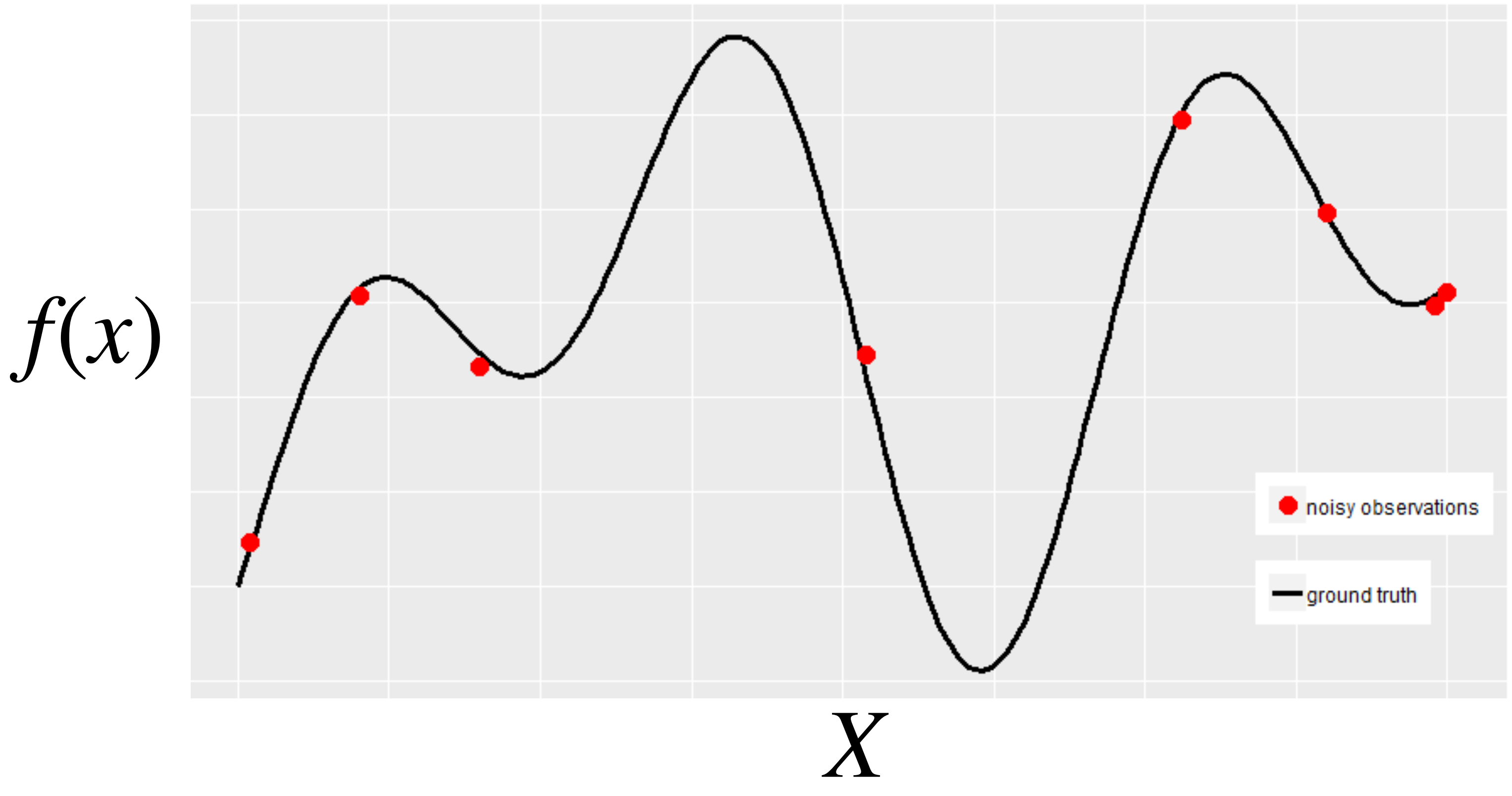


Estimation of Distribution Algorithms
(Bayesian Optimization)
Function $f(x)$
does not change with x (static)
Goal: finding the best x
Process: learn and improve decisions iteratively
Uncertainty: unknown, noisy $f(x)$
From an unknown dist. D1
From an unknown dist. D2
From an unknown dist. D3
From an unknown dist. D4
x :Arm 1
x : Arm 2
x :Arm 3
x :Arm 4
Observation:
$f(x)$ =
Reinforcement Learning
Function $f(x)$ changes with x (dynamic)
Goal: finding a policy that gives the best x at each state of $f(x)$
Process: learn and improve decisions iteratively
Uncertainty: unknown, noisy $f(x)$
x : Go left
x : Stay
x : Go right
Observation:
1) new state of f after applying x
2) How good x was

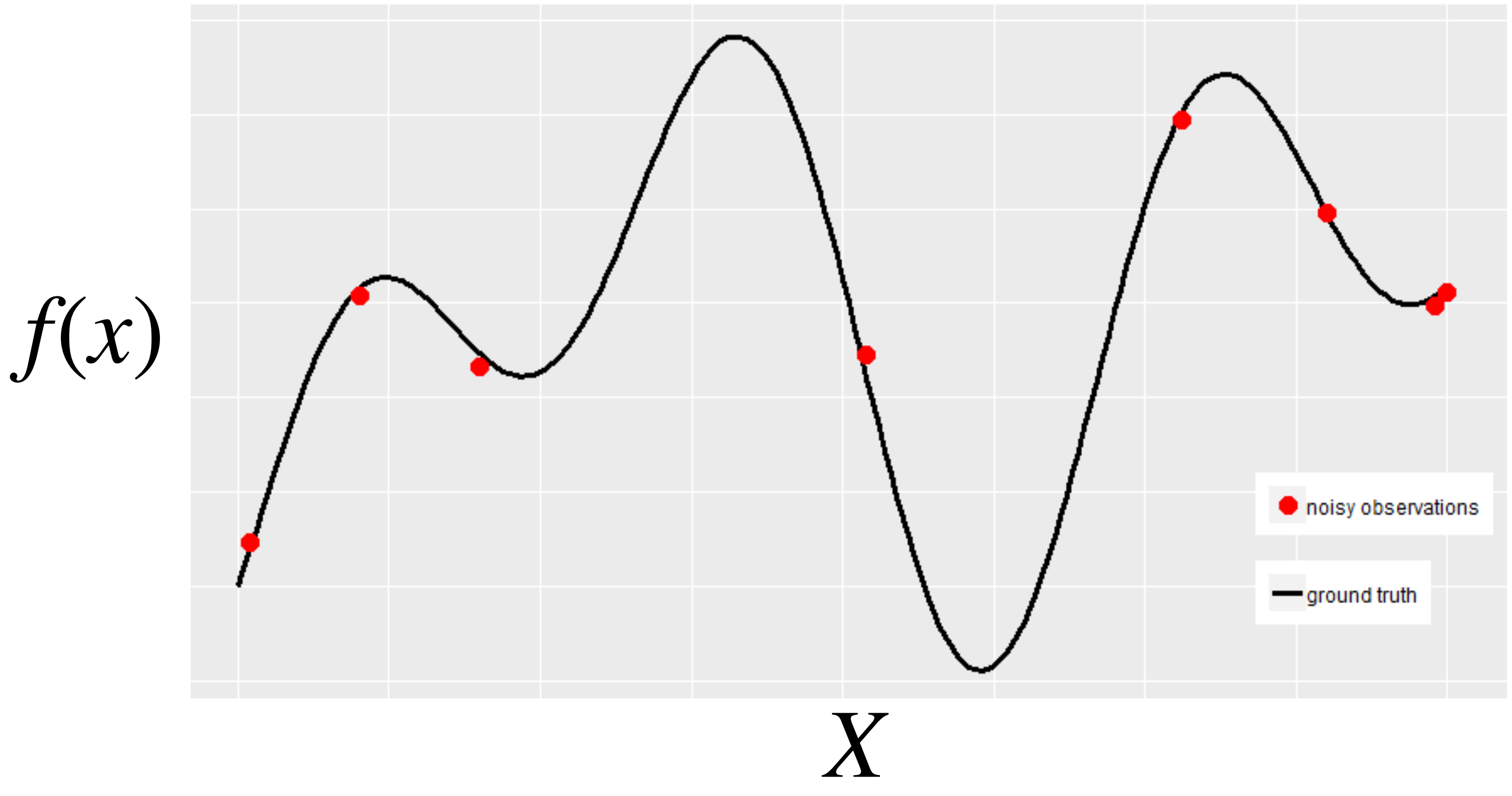
Bayesian Optimization

Bayesian Optimization

Bayesian Optimization

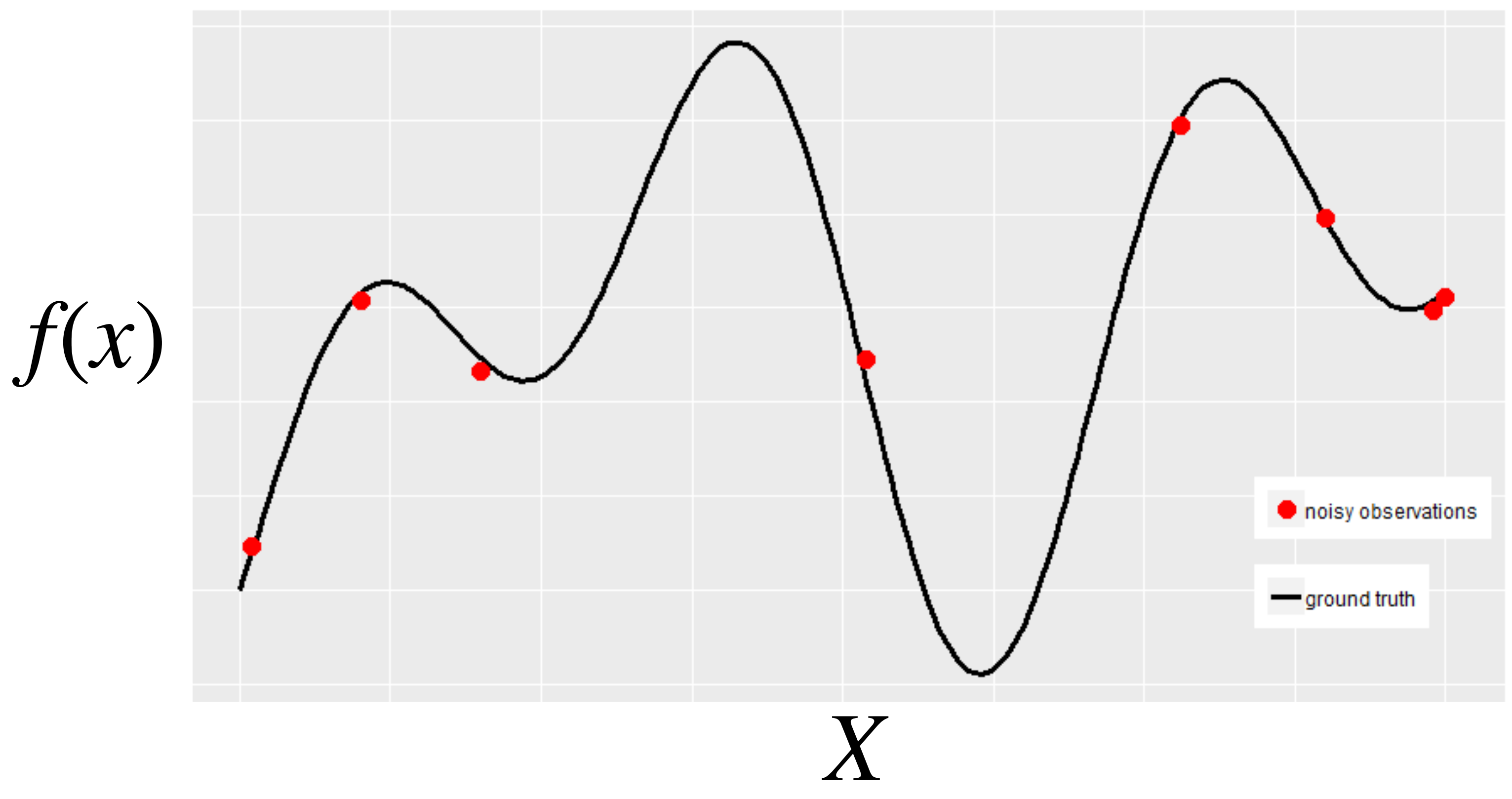


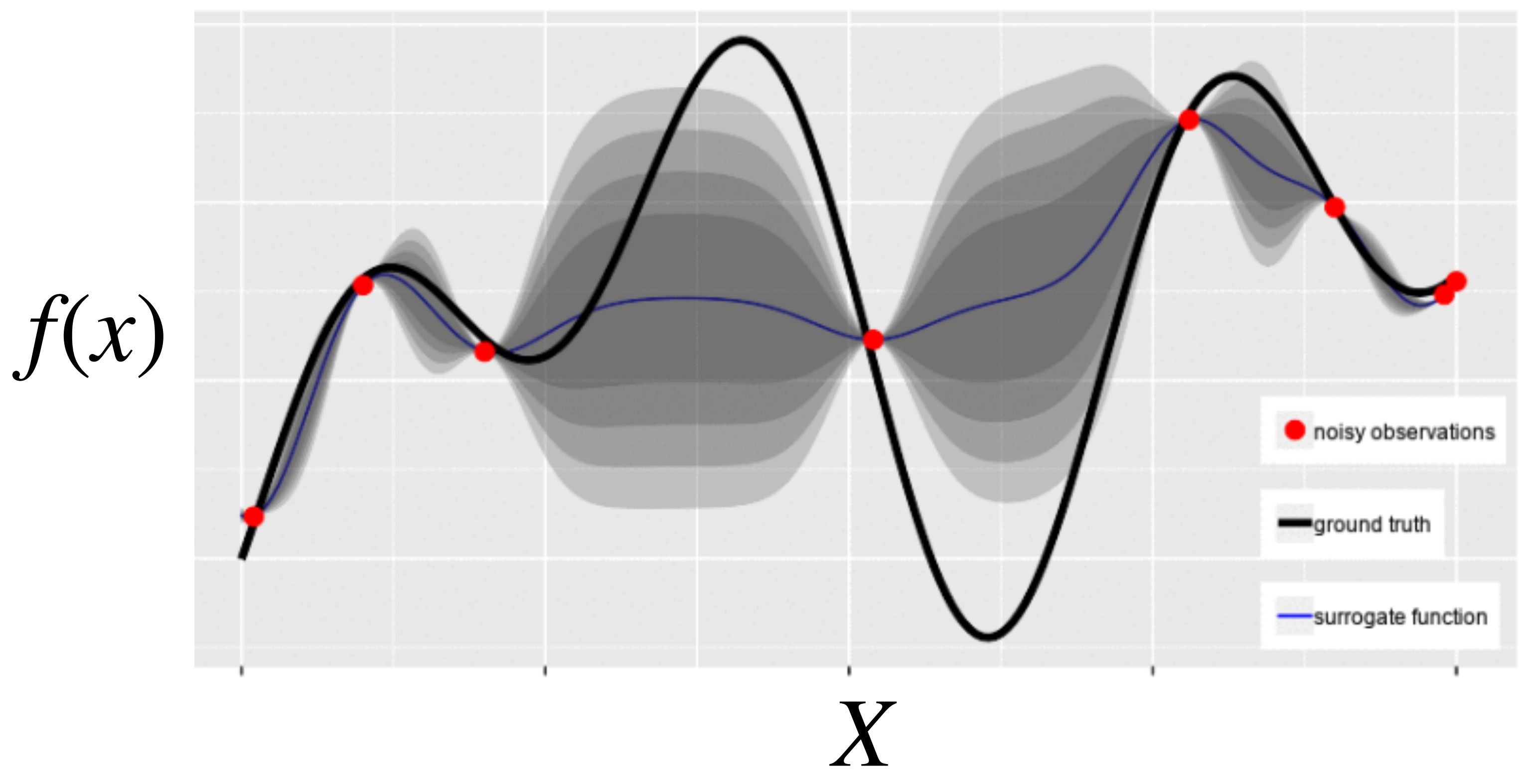
Bayesian Optimization



1) initial sample

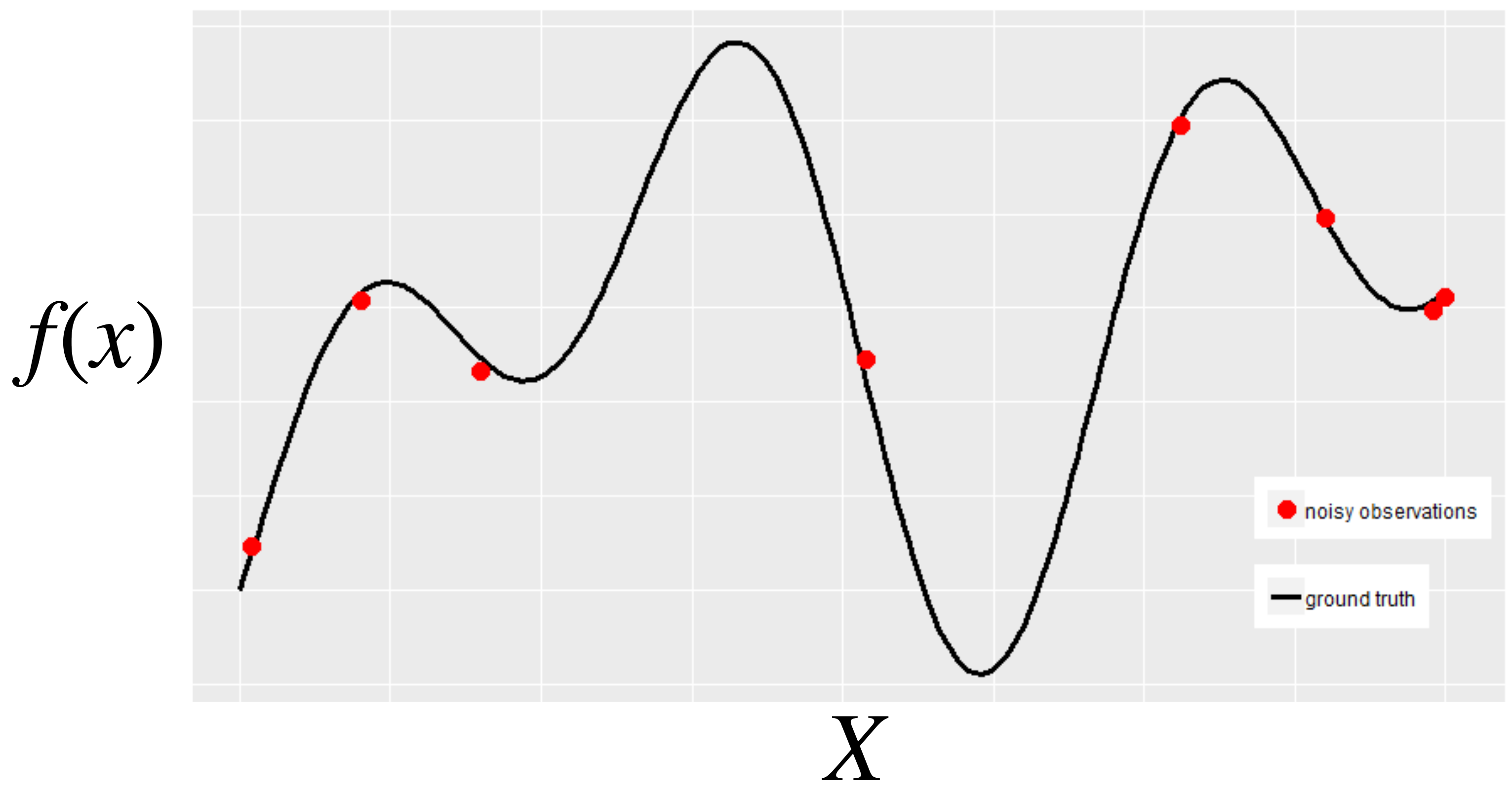
Bayesian Optimization

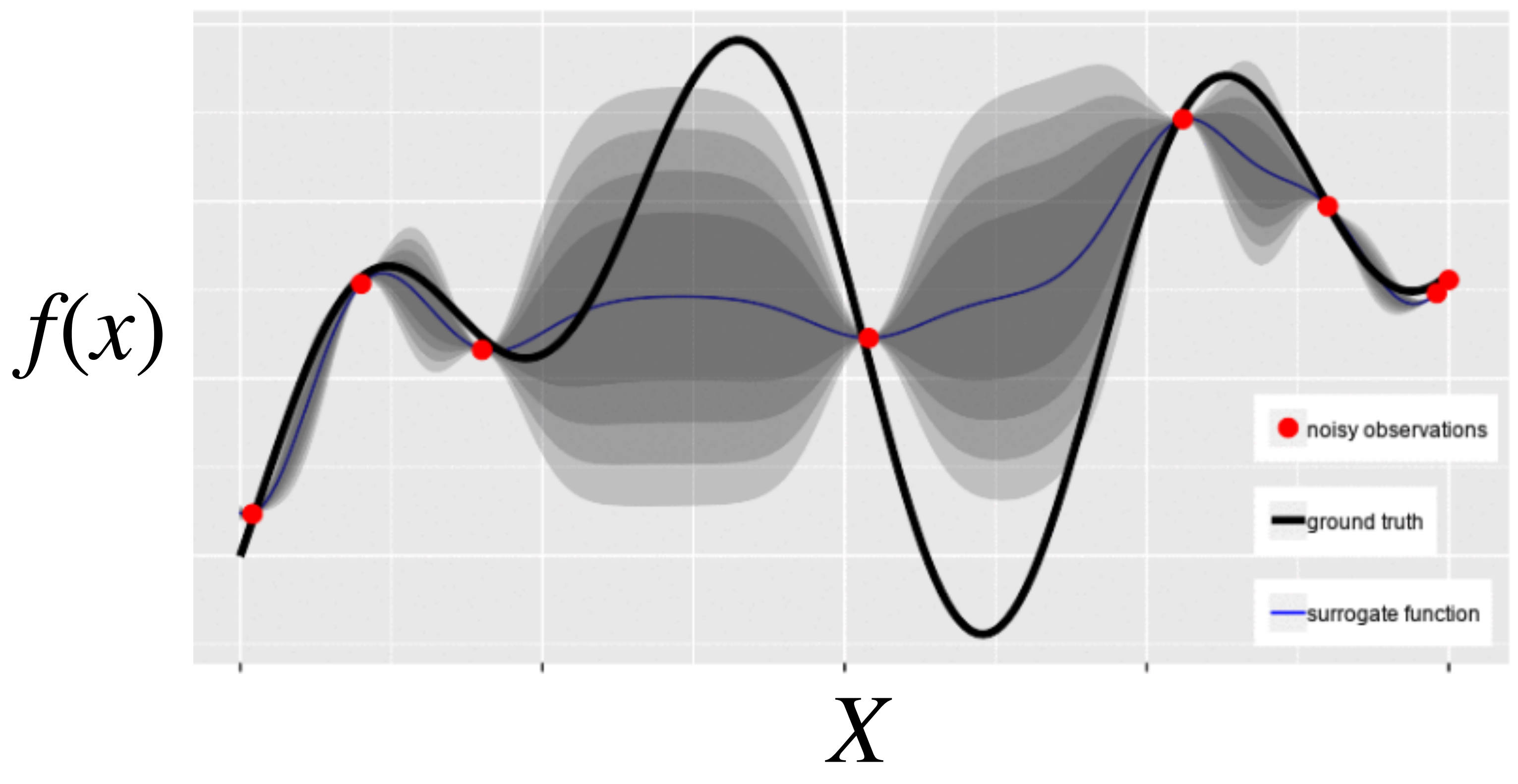


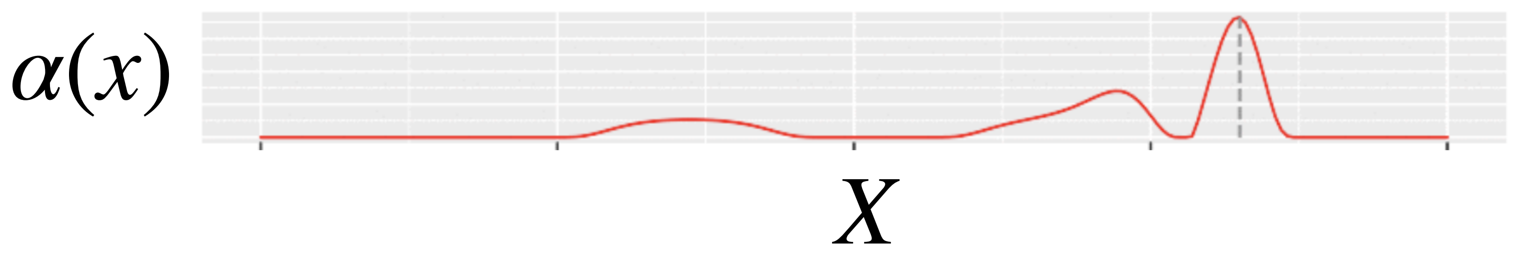


1) initial sample
2) fit a surrogate model given samples

Bayesian Optimization



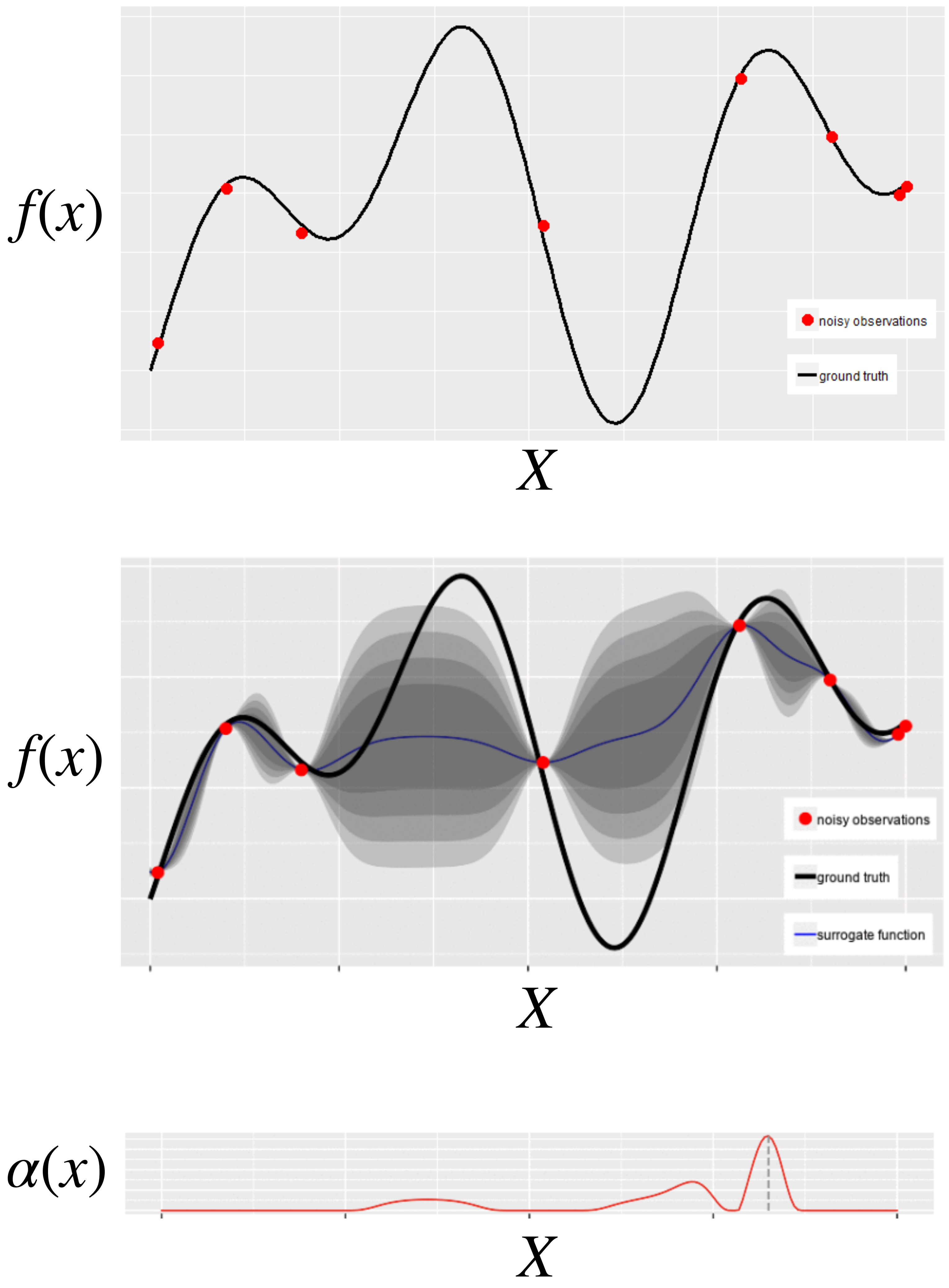




1) initial sample
2) fit a surrogate model given samples
3) obtain an acquisition function $\alpha(x)$

MITSUBISHI ELECTRIC UNIVERSITY OF ALBERTA amii

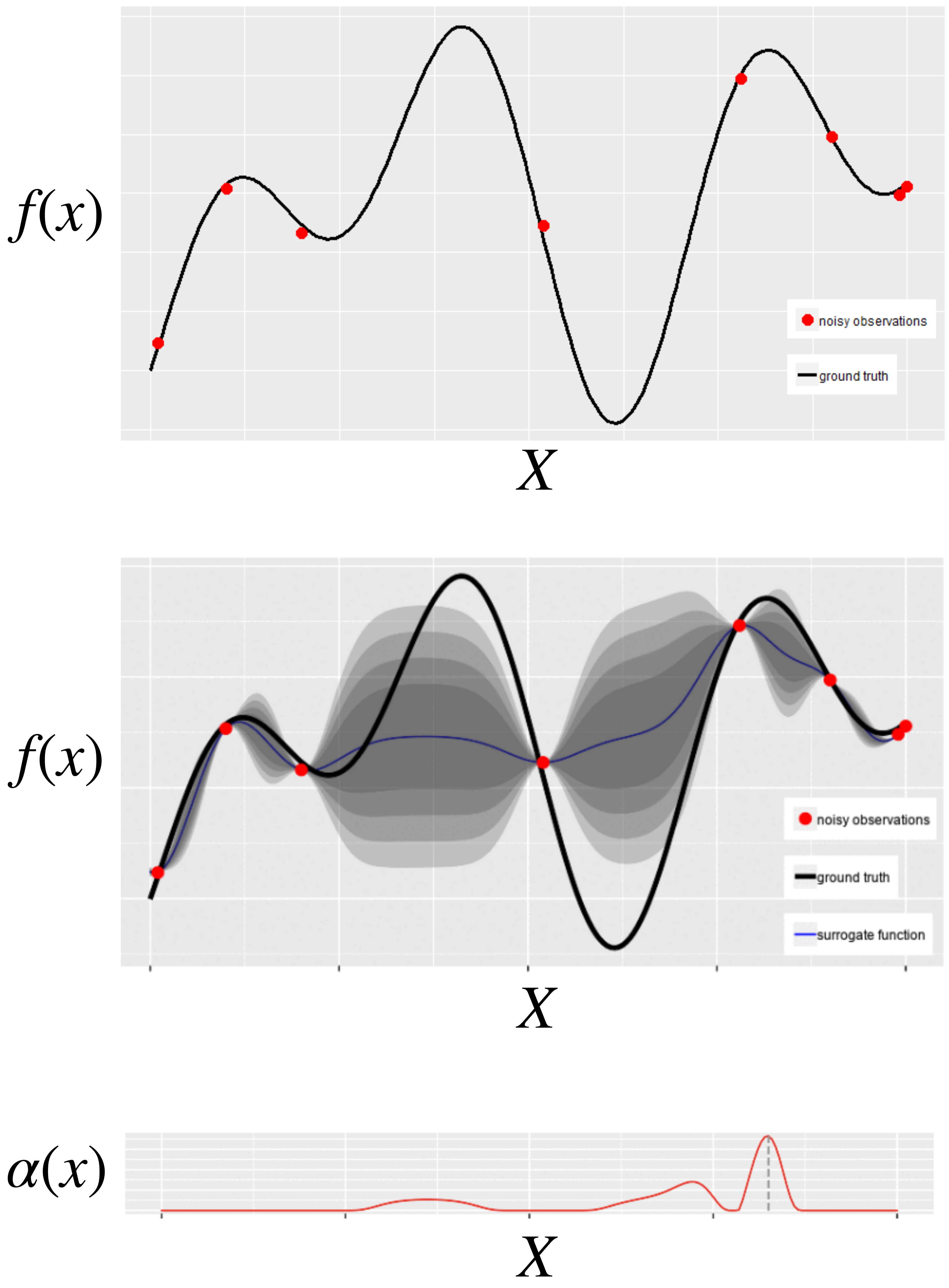
Bayesian Optimization



1) initial sample
2) fit a surrogate model given samples
3) obtain an acquisition function $\alpha(x)$
4) optimize $\alpha(x)$

$$x_{next} = \arg \underset{x \in X}{max} \ \alpha(x)$$

Bayesian Optimization



1) initial sample
2) fit a surrogate model given samples
3) obtain an acquisition function $\alpha(x)$
4) optimize $\alpha(x)$

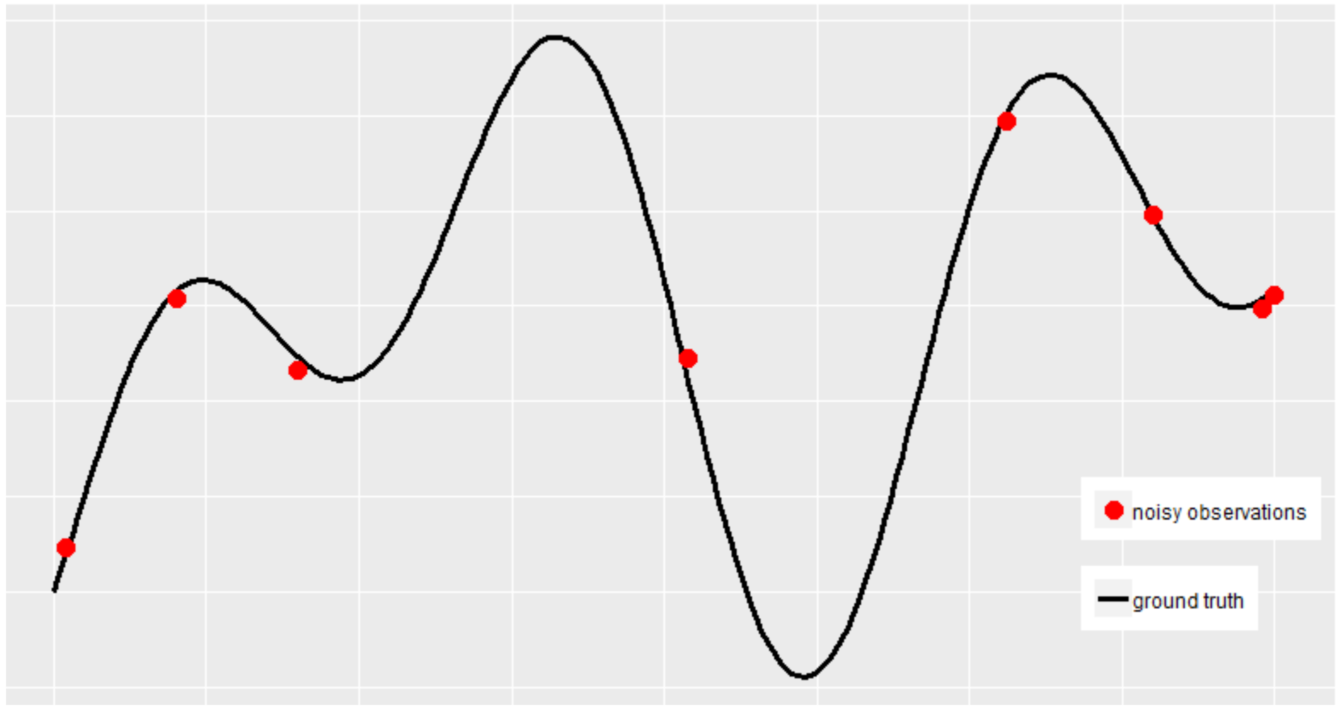
$$x_{next} = \underset{x \in X}{\arg\,max}\ \alpha(x)$$

5) sample x_{next}

MITSUBISHI ELECTRIC UNIVERSITY OF ALBERTA amii

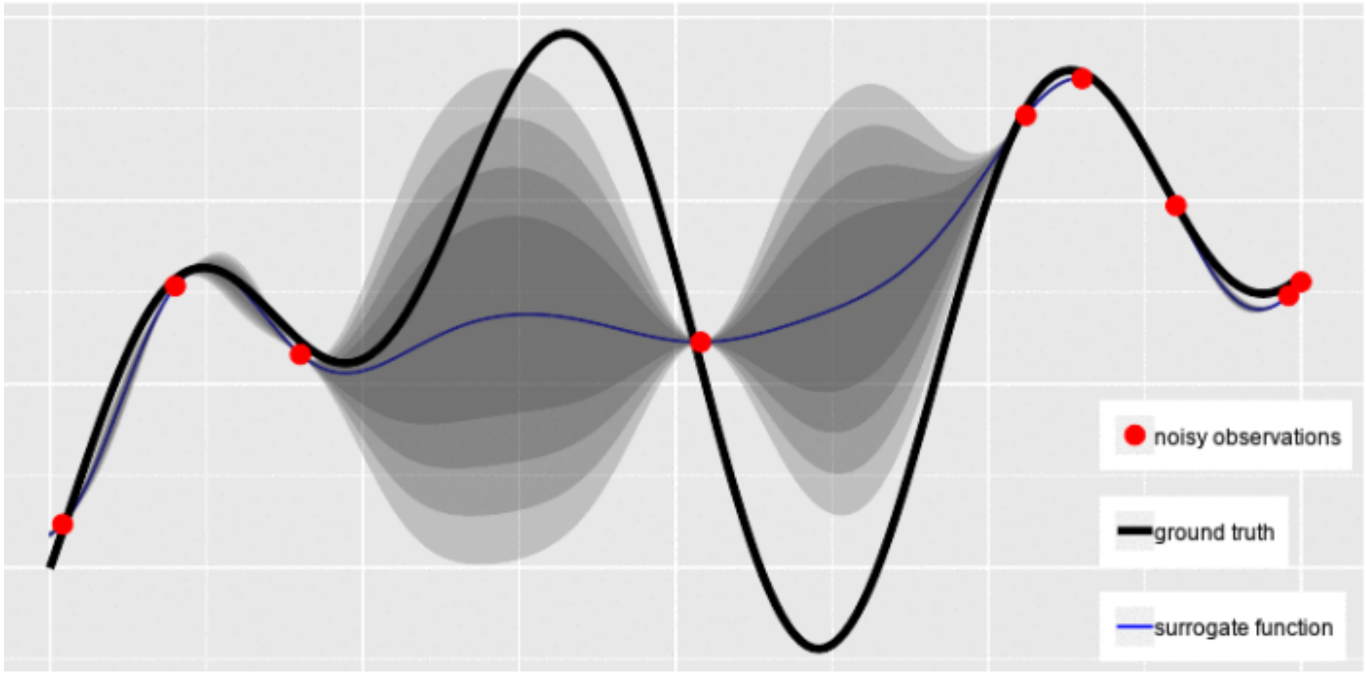
Bayesian Optimization

$f(x)$



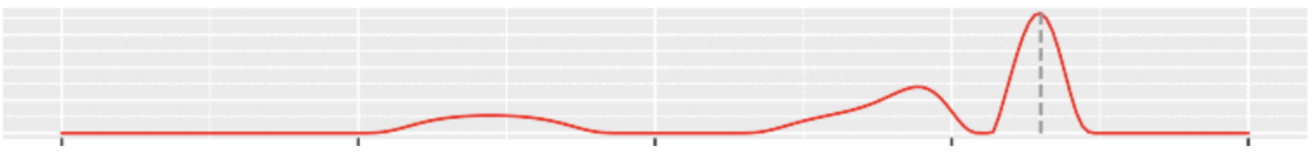
X

$f(x)$



X

$\alpha(x)$



X

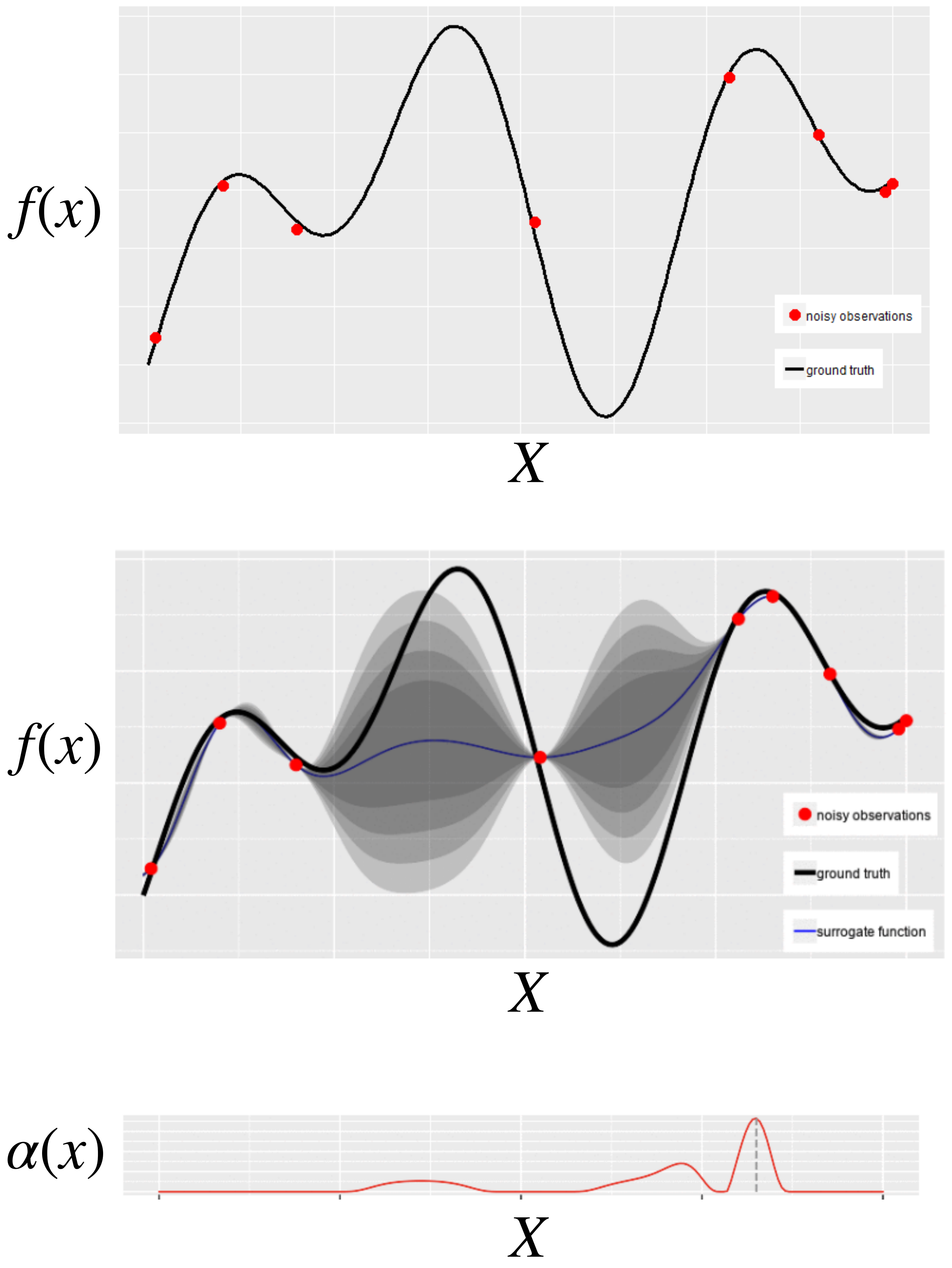
1) initial sample
2) fit a surrogate model given samples
3) obtain an acquisition function $\alpha(x)$
4) optimize $\alpha(x)$

$$x_{next} = \arg \max_{x \in X} \alpha(x)$$

5) sample x_{next}
6) update surrogate model

MITSUBISHI ELECTRIC UNIVERSITY OF ALBERTA amii

Bayesian Optimization



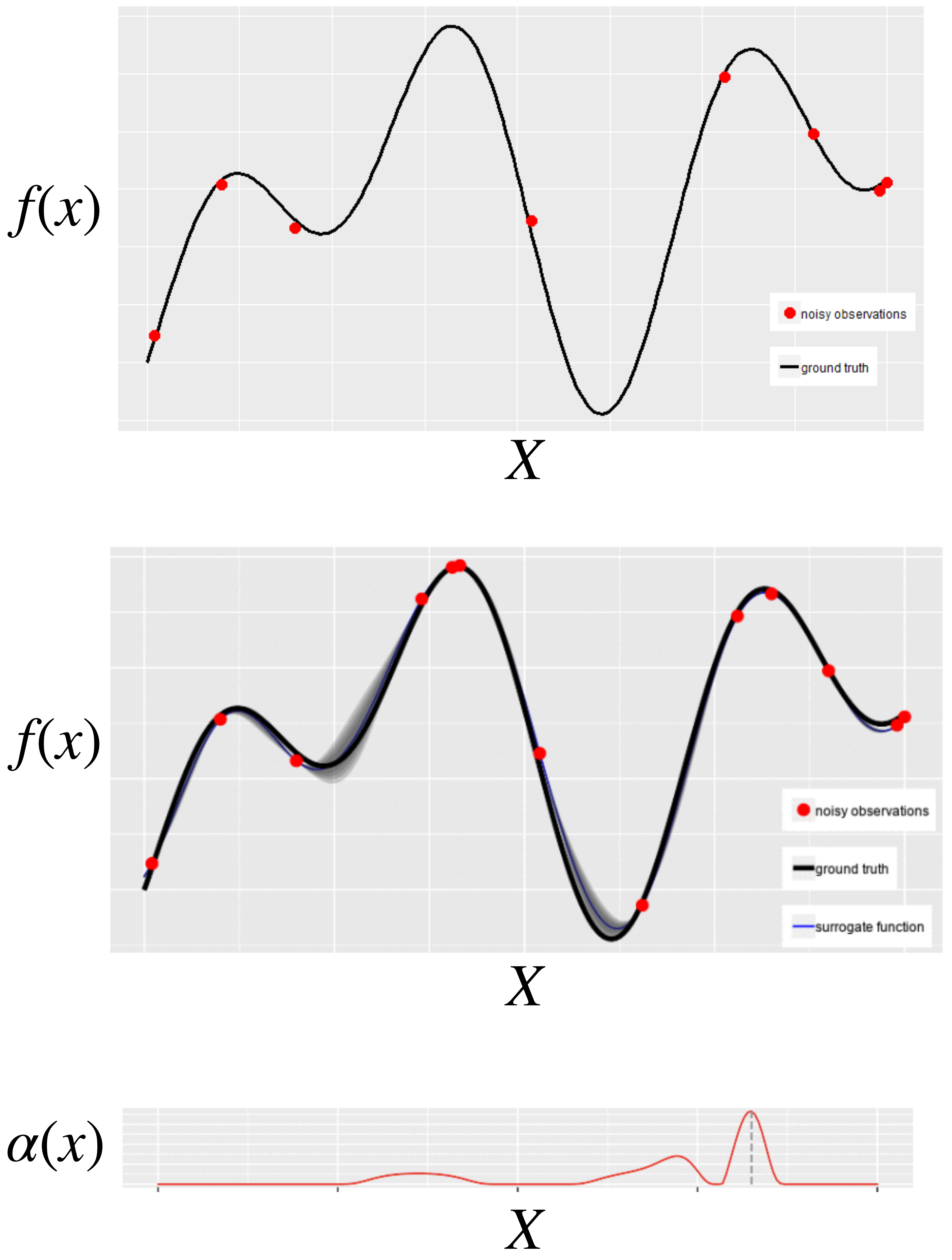
1) initial sample
2) fit a surrogate model given samples
3) obtain an acquisition function $\alpha(x)$
4) optimize $\alpha(x)$

$$x_{next} = \arg \max_{x \in X} \alpha(x)$$

5) sample x_{next}
6) update surrogate model
7) iterate from 3

MITSUBISHI ELECTRIC UNIVERSITY OF ALBERTA amii

Bayesian Optimization



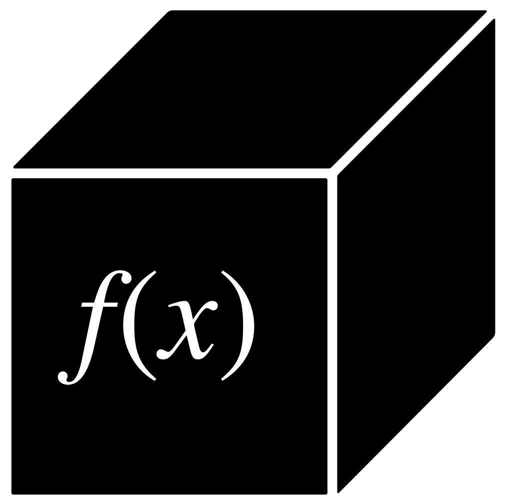
1) initial sample
2) fit a surrogate model given samples
3) obtain an acquisition function $\alpha(x)$
4) optimize $\alpha(x)$

$$x_{next} = \arg \underset{x \in X}{max}\ \alpha(x)$$

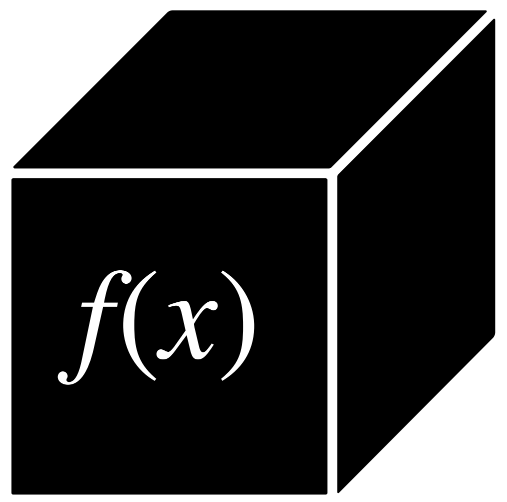
5) sample x_{next}
6) update surrogate model
7) iterate from 3

MITSUBISHI ELECTRIC UNIVERSITY OF ALBERTA amii

Bayesian Optimization

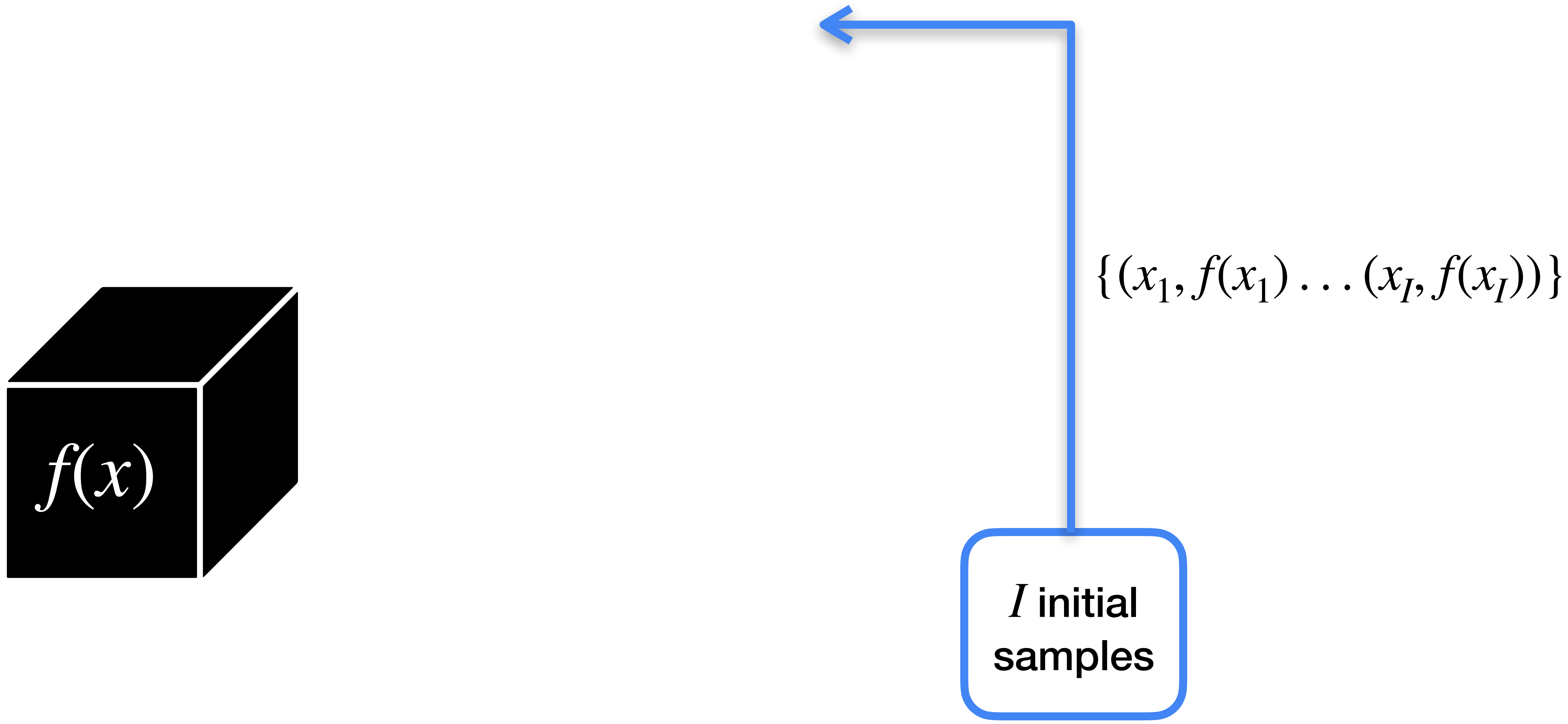


Bayesian Optimization

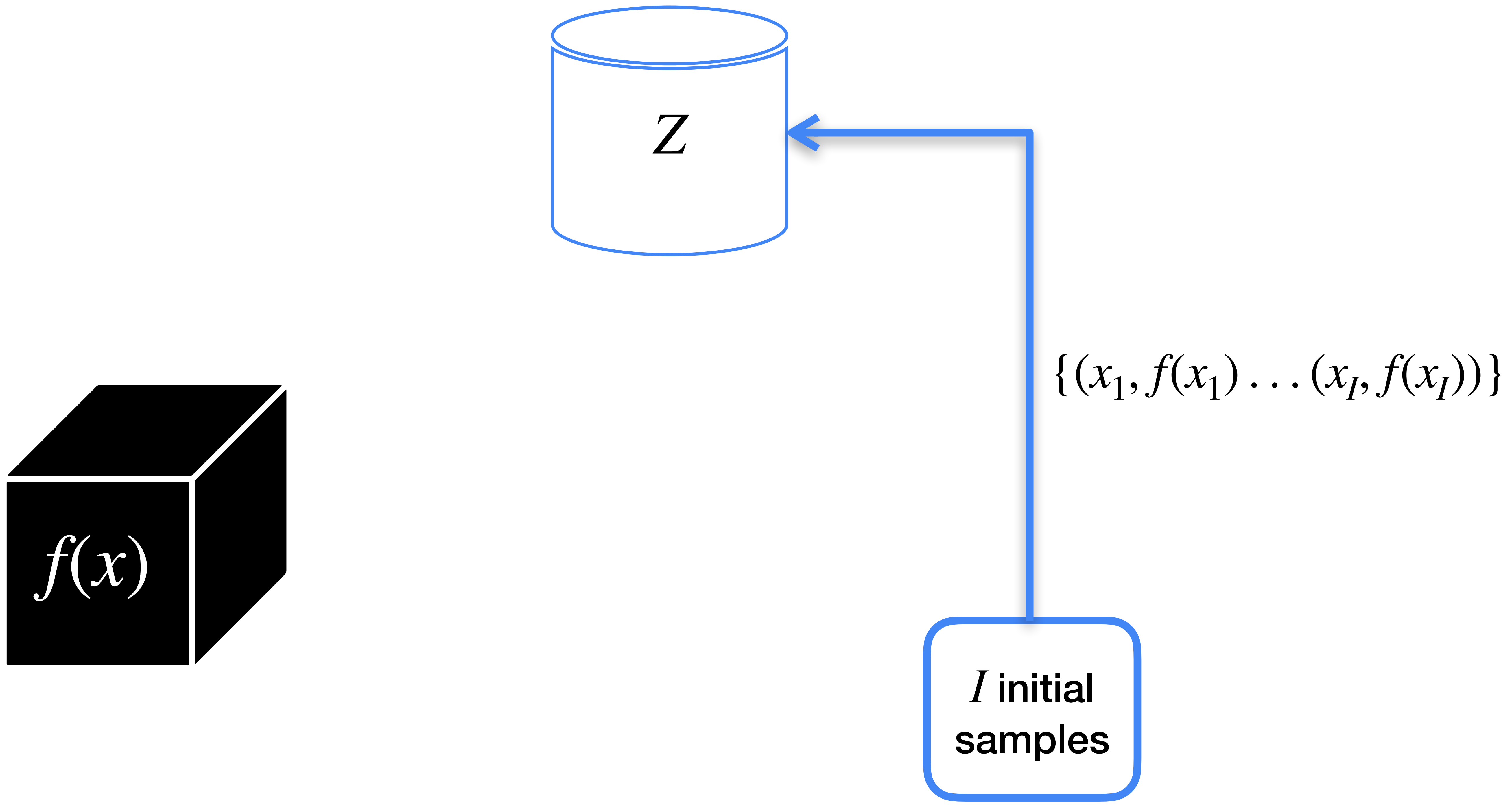


I initial samples

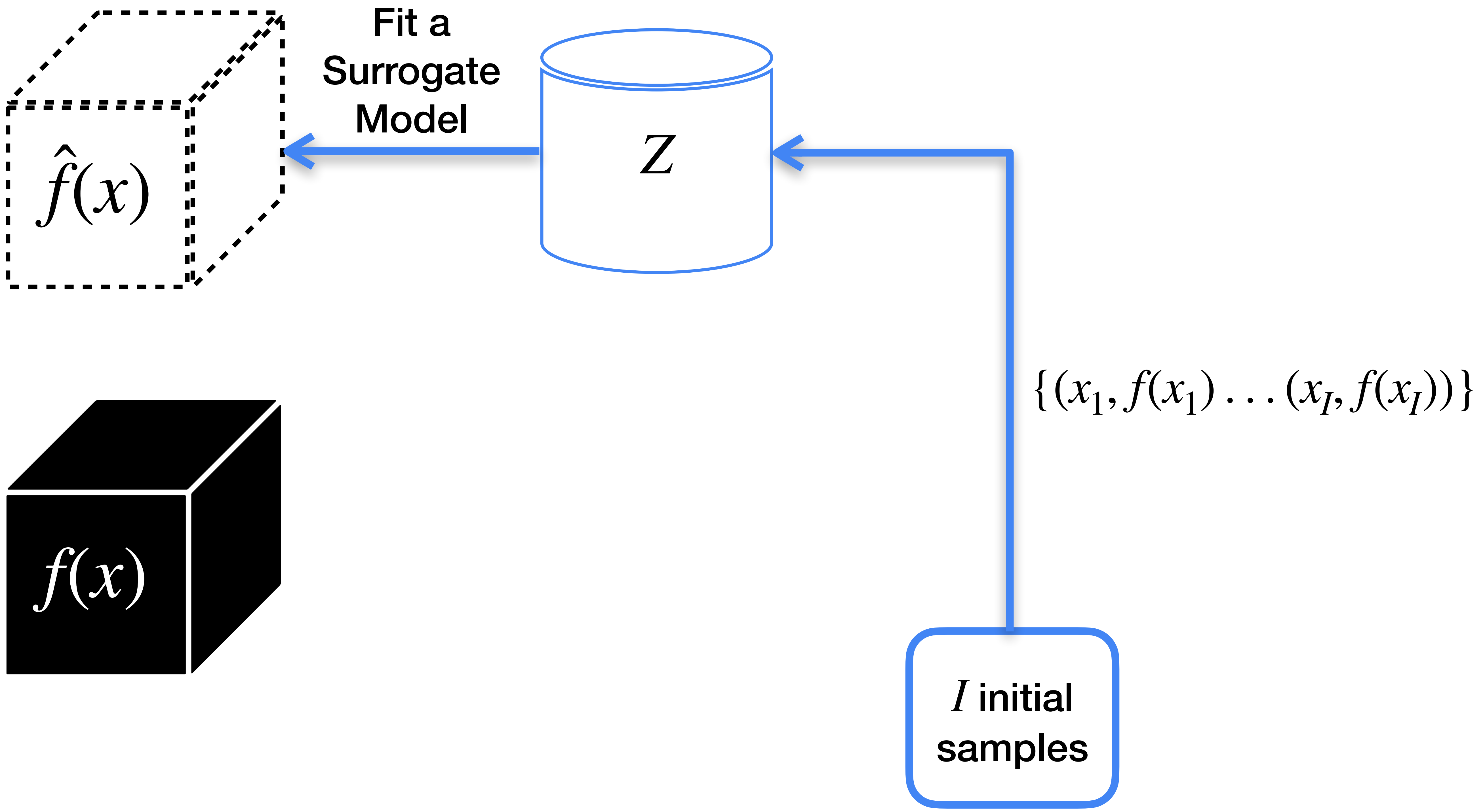
Bayesian Optimization



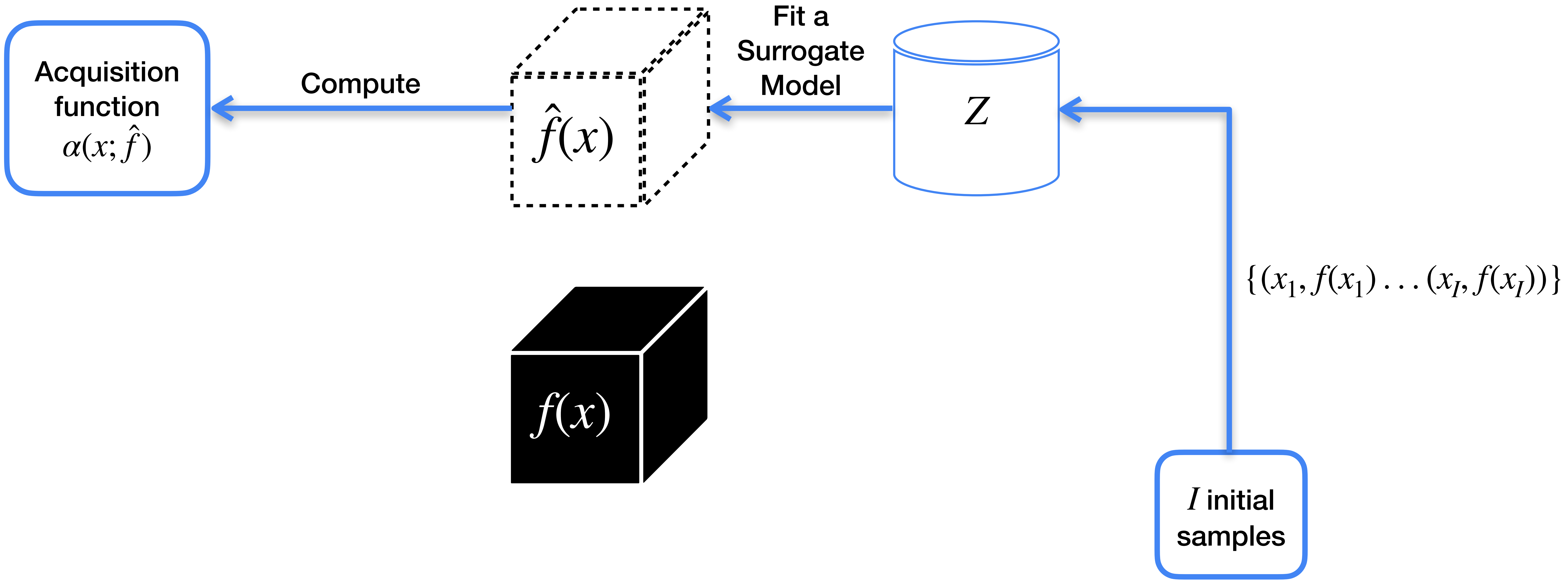
Bayesian Optimization



Bayesian Optimization

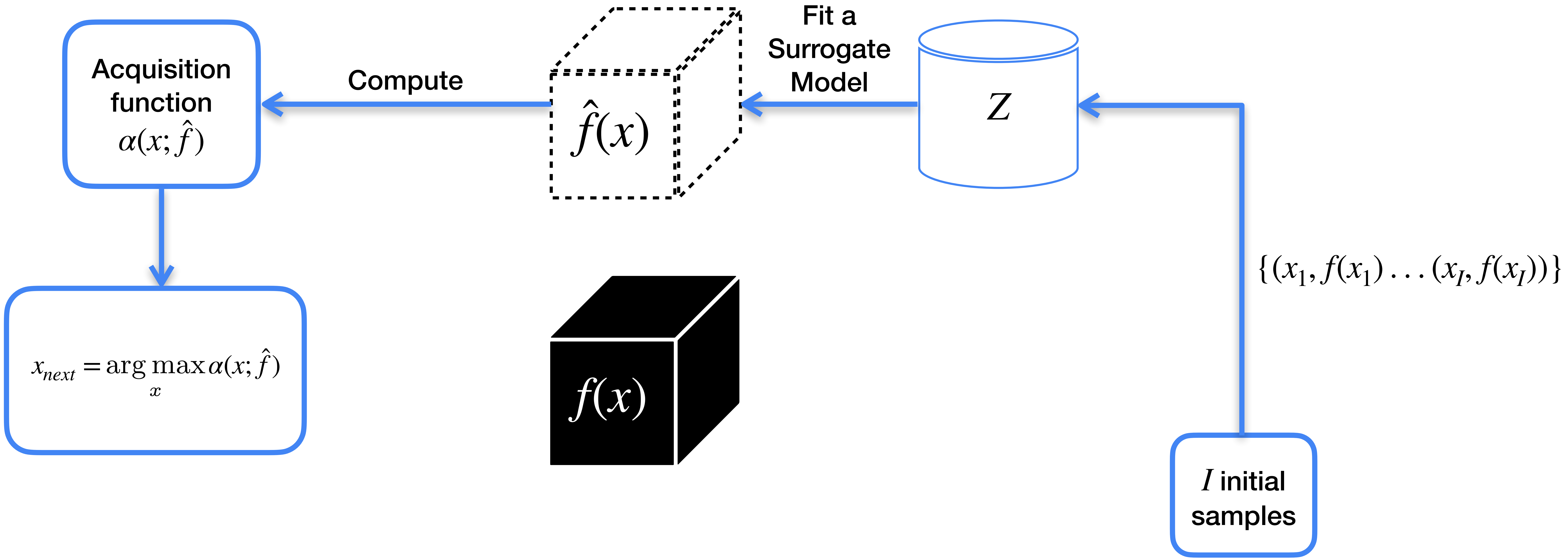


Bayesian Optimization



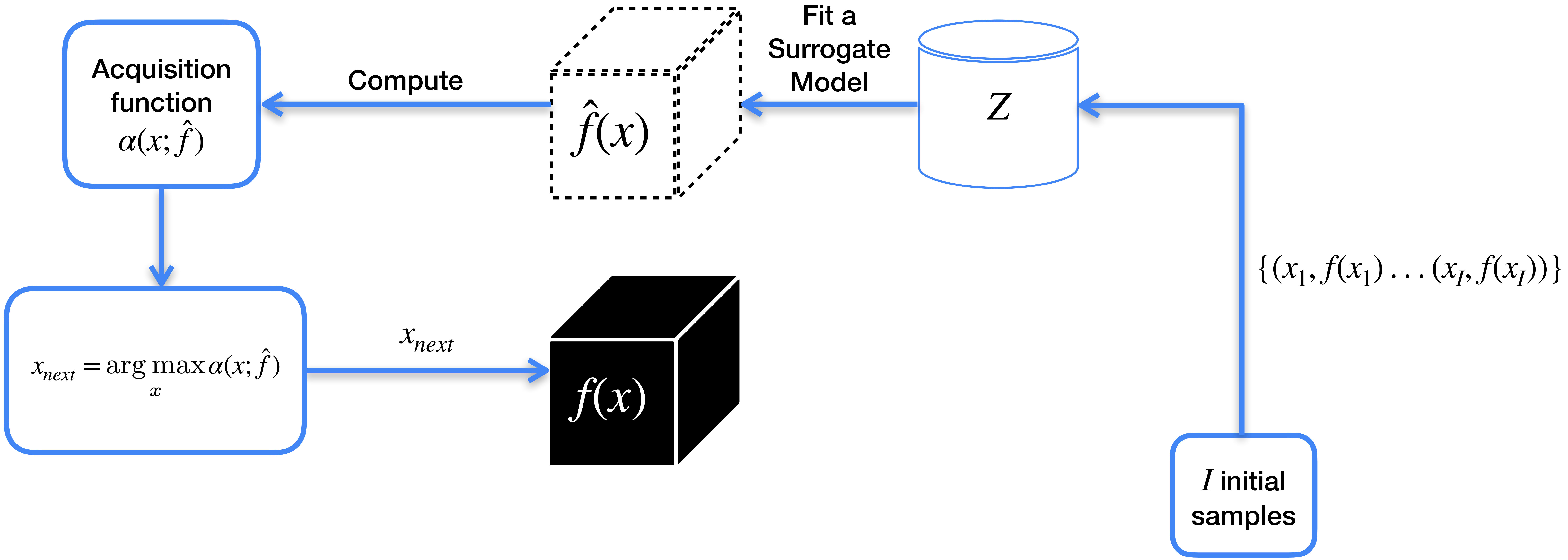
MITSUBISHI ELECTRIC UNIVERSITY OF ALBERTA amii

Bayesian Optimization

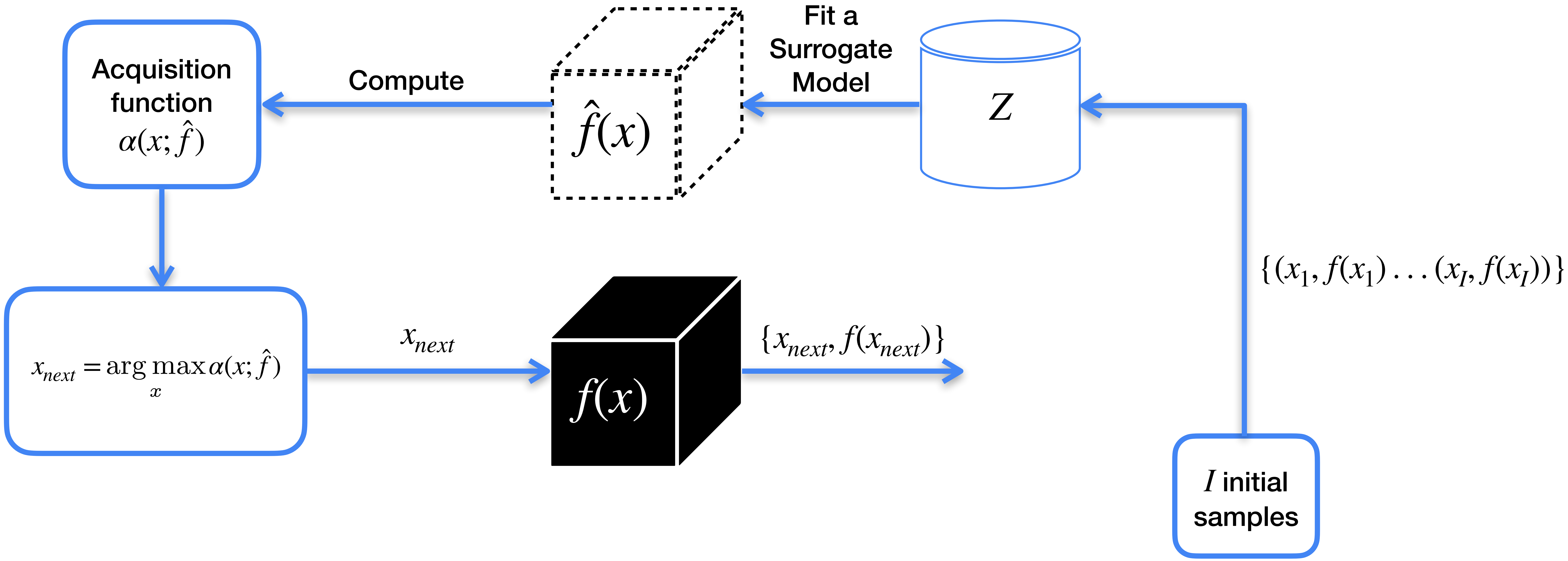


MITSUBISHI ELECTRIC UNIVERSITY OF ALBERTA amii

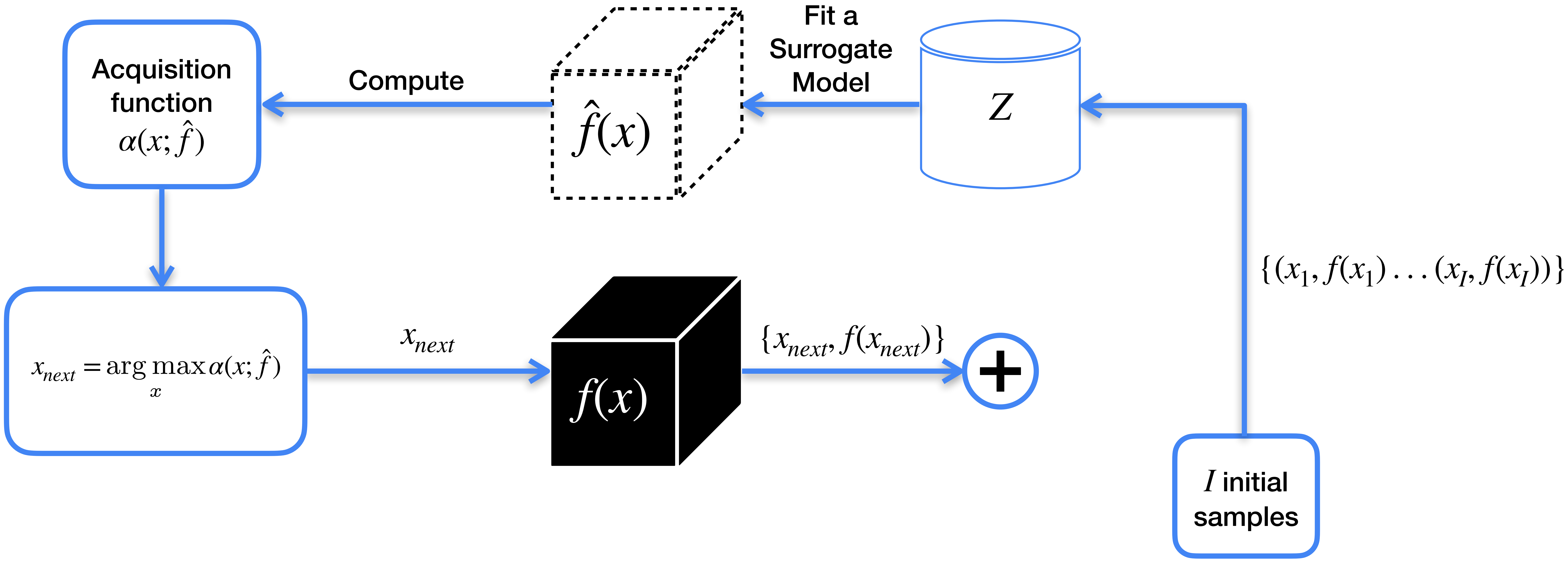
Bayesian Optimization



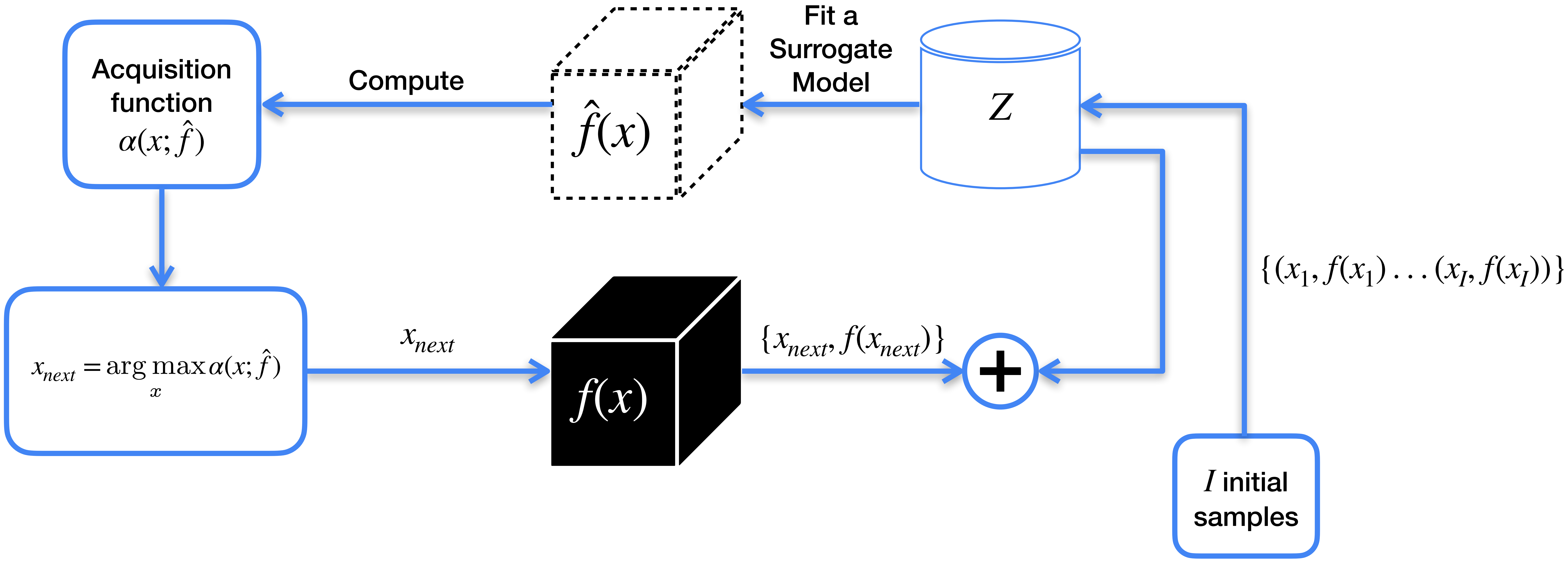
Bayesian Optimization



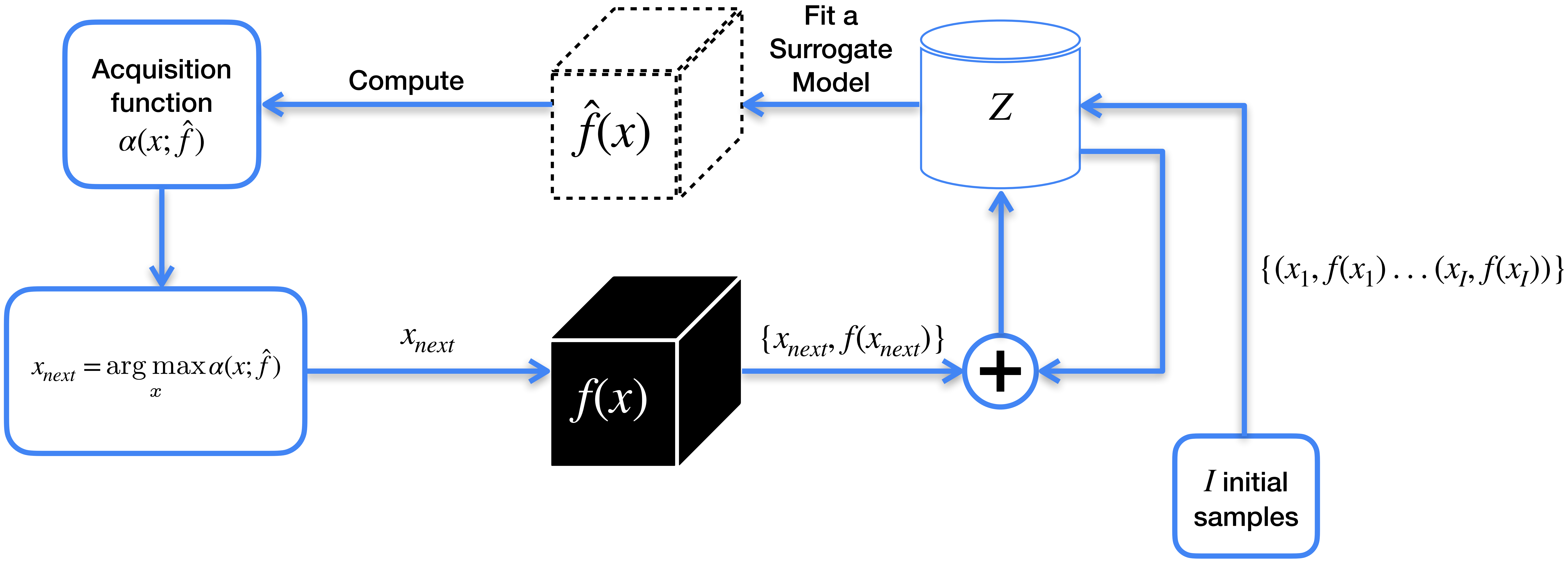
Bayesian Optimization



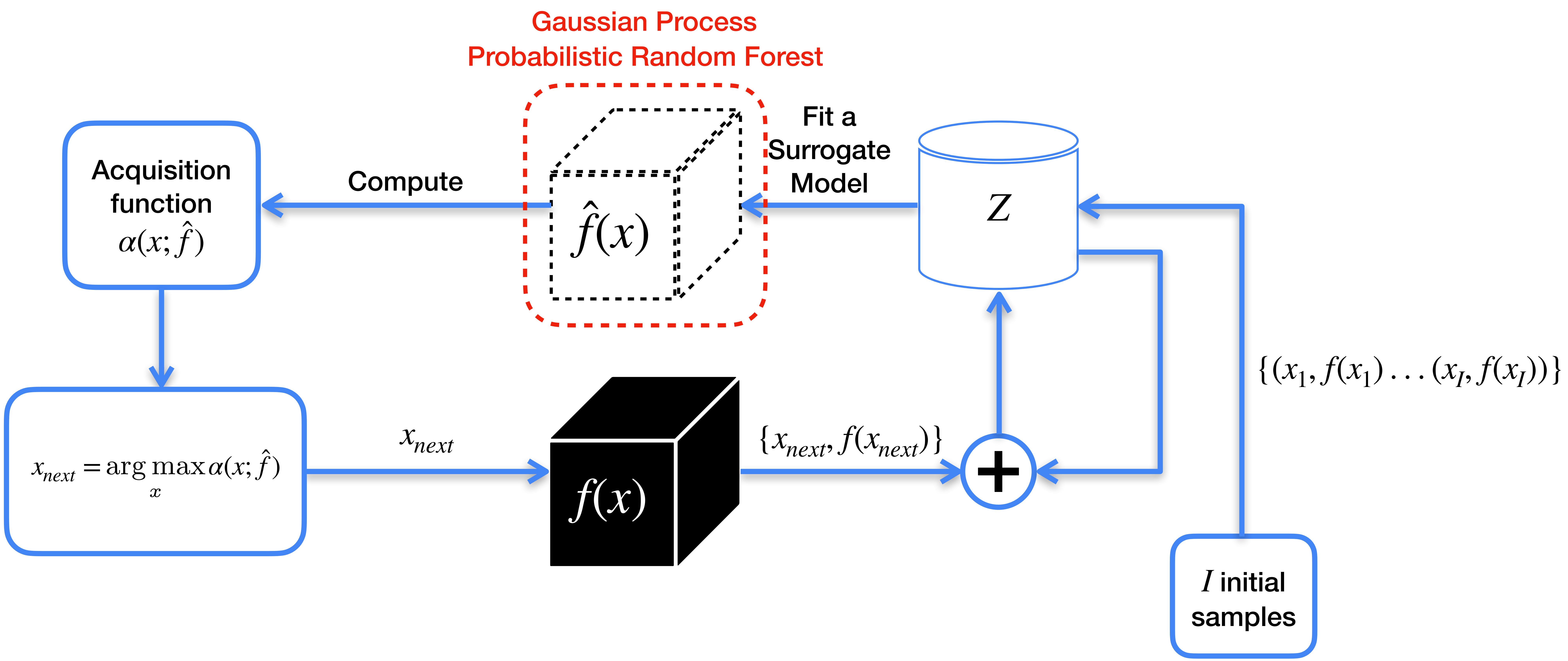
Bayesian Optimization



Bayesian Optimization

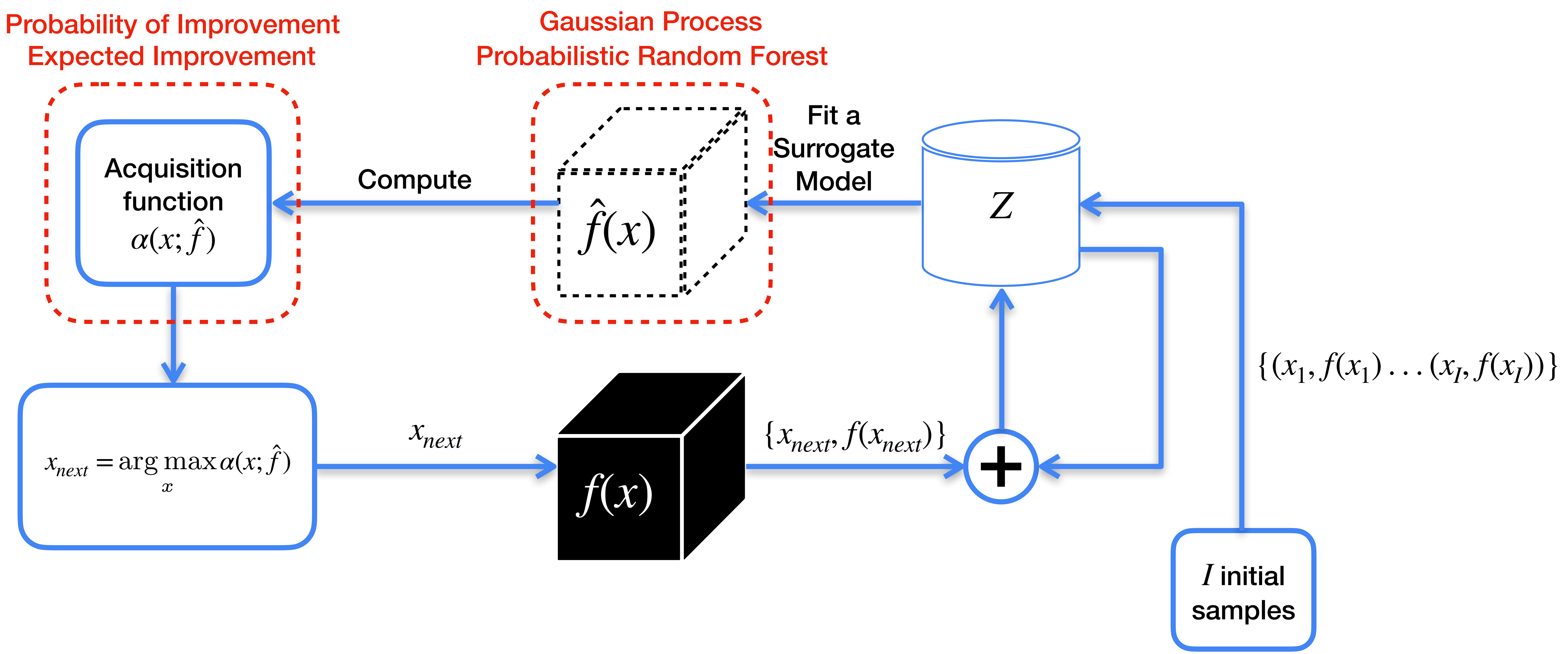


Bayesian Optimization



MITSUBISHI ELECTRIC UNIVERSITY OF ALBERTA amii

Bayesian Optimization



MITSUBISHI ELECTRIC UNIVERSITY OF ALBERTA amii

Bayesian Optimization

Real-World Applications

Bayesian Optimization

Real-World Applications

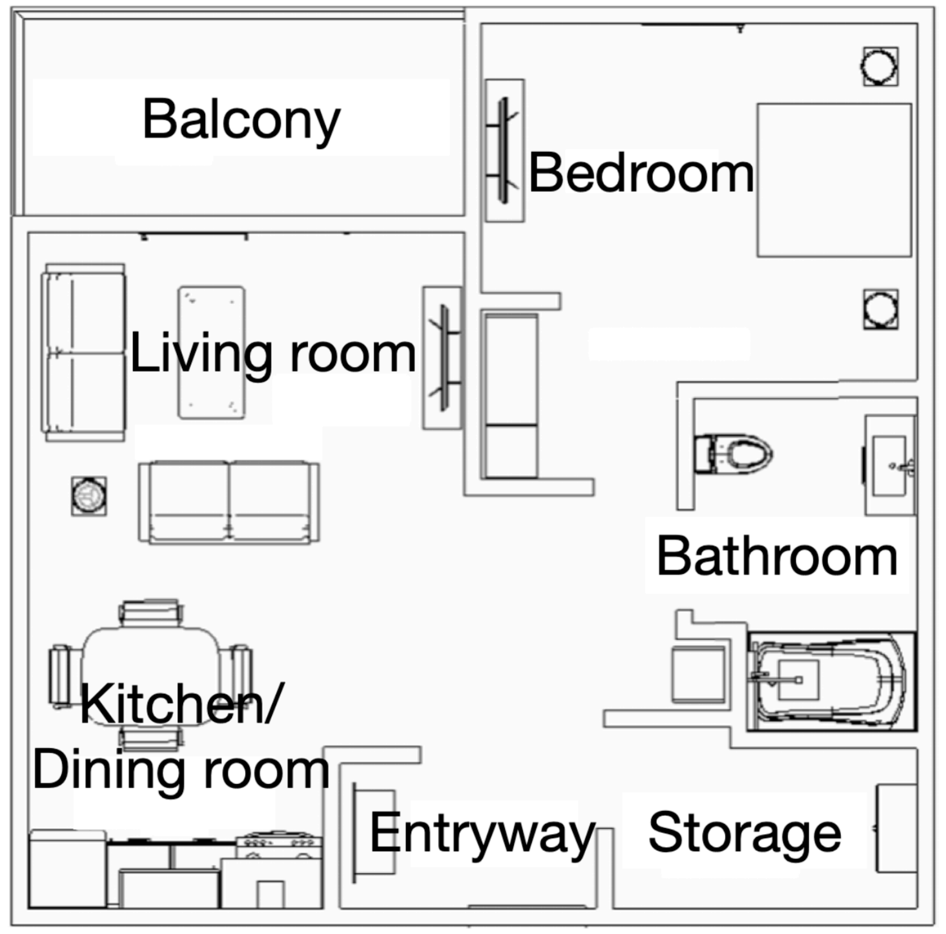
System Design Optimization

Bayesian Optimization

Real-World Applications

System Design Optimization

Sensor Placement Optimization



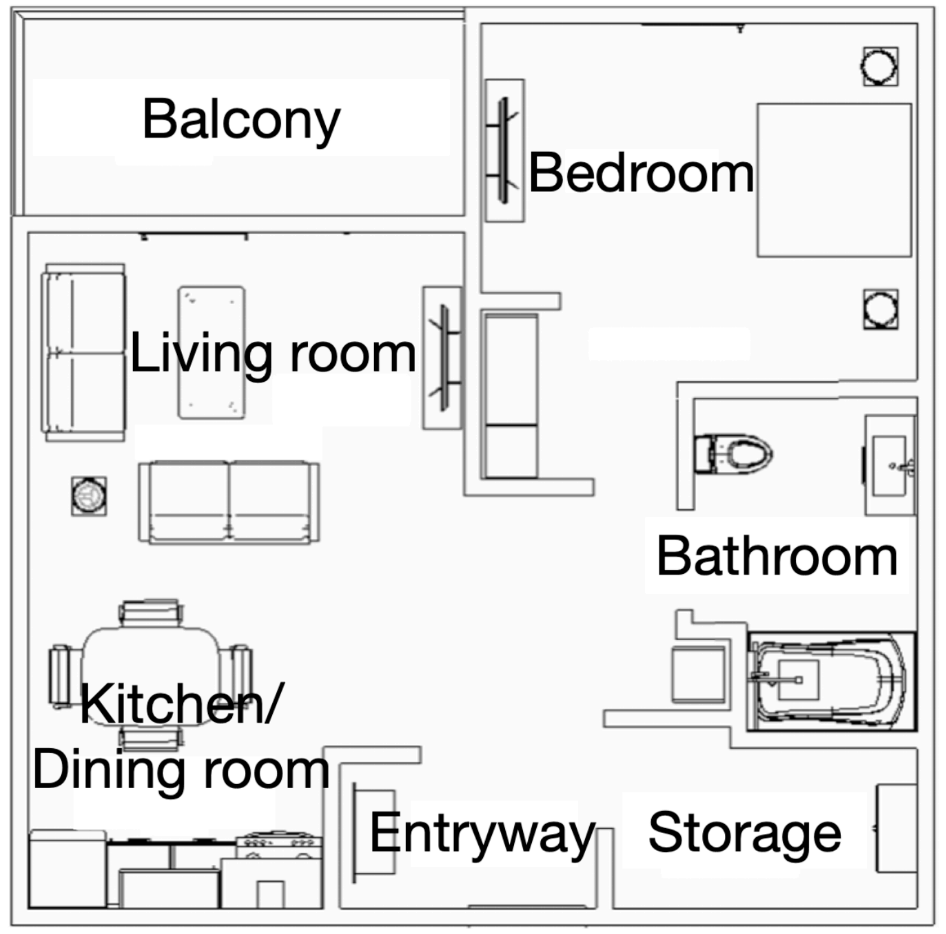
Golestan, Shadan, Omid Ardakanian, and Pierre Boulanger. "Grey-Box Bayesian Optimization for Sensor Placement in Assisted Living Environments." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. No. 20. 2024.

Bayesian Optimization

Real-World Applications

System Design Optimization

Sensor Placement Optimization



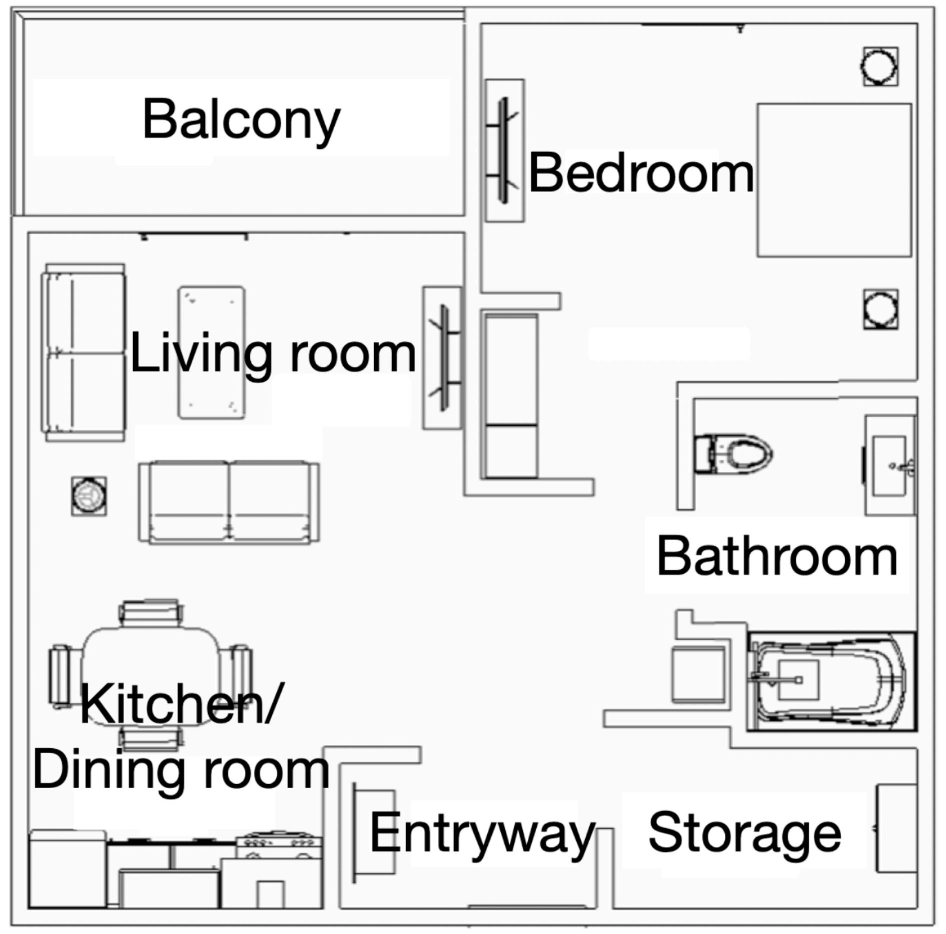
Golestan, Shadan, Omid Ardakanian, and Pierre Boulanger. "Grey-Box Bayesian Optimization for Sensor Placement in Assisted Living Environments." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. No. 20. 2024.

Bayesian Optimization

Real-World Applications

System Design Optimization

Sensor Placement Optimization



Golestan, Shadan, Omid Ardakanian, and Pierre Boulanger. "Grey-Box Bayesian Optimization for Sensor Placement in Assisted Living Environments." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. No. 20. 2024.

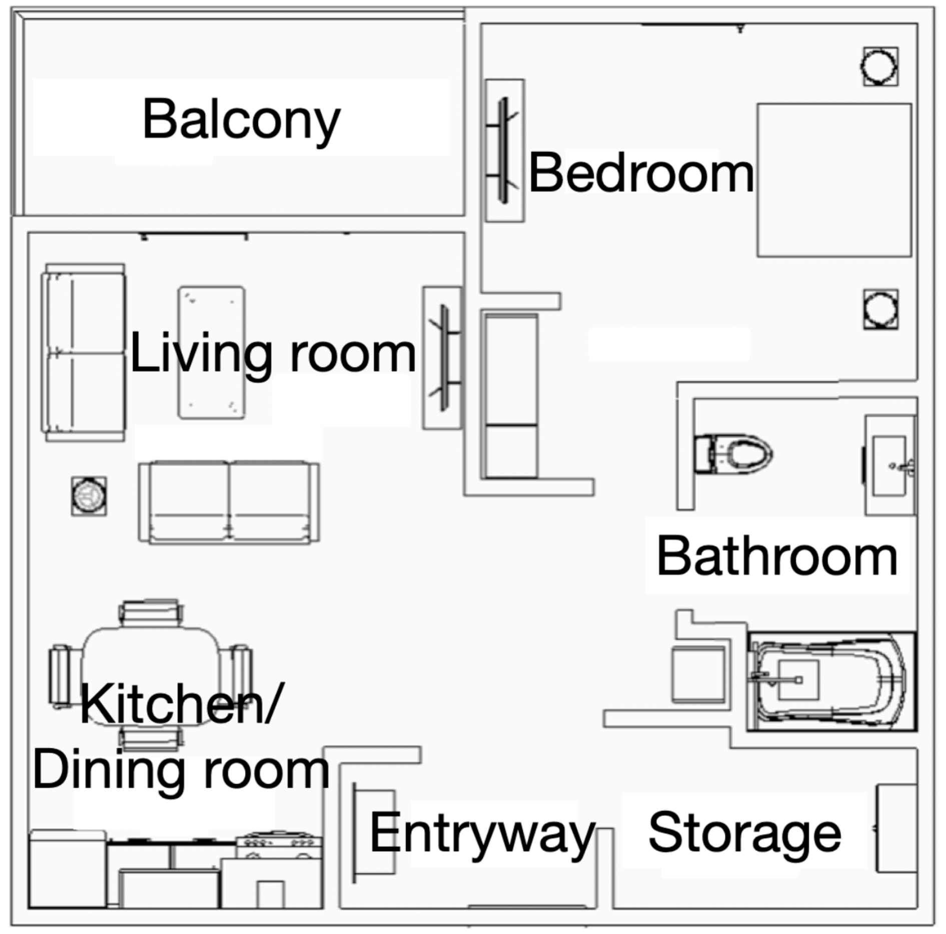
Bayesian Optimization

Real-World Applications

System Design Optimization

AutoML (Hyperparameter Optimization)

Sensor Placement Optimization



Golestan, Shadan, Omid Ardakanian, and Pierre Boulanger. "Grey-Box Bayesian Optimization for Sensor Placement in Assisted Living Environments." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. No. 20. 2024.

Bayesian Optimization

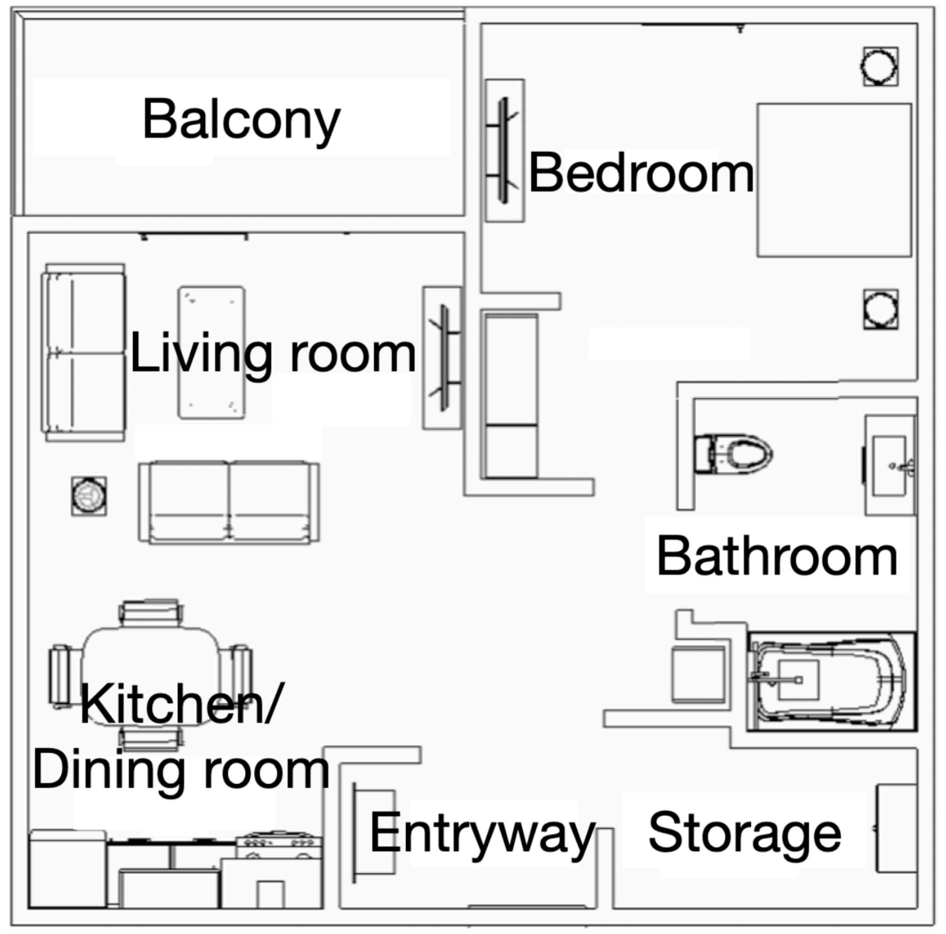
Real-World Applications

System Design Optimization

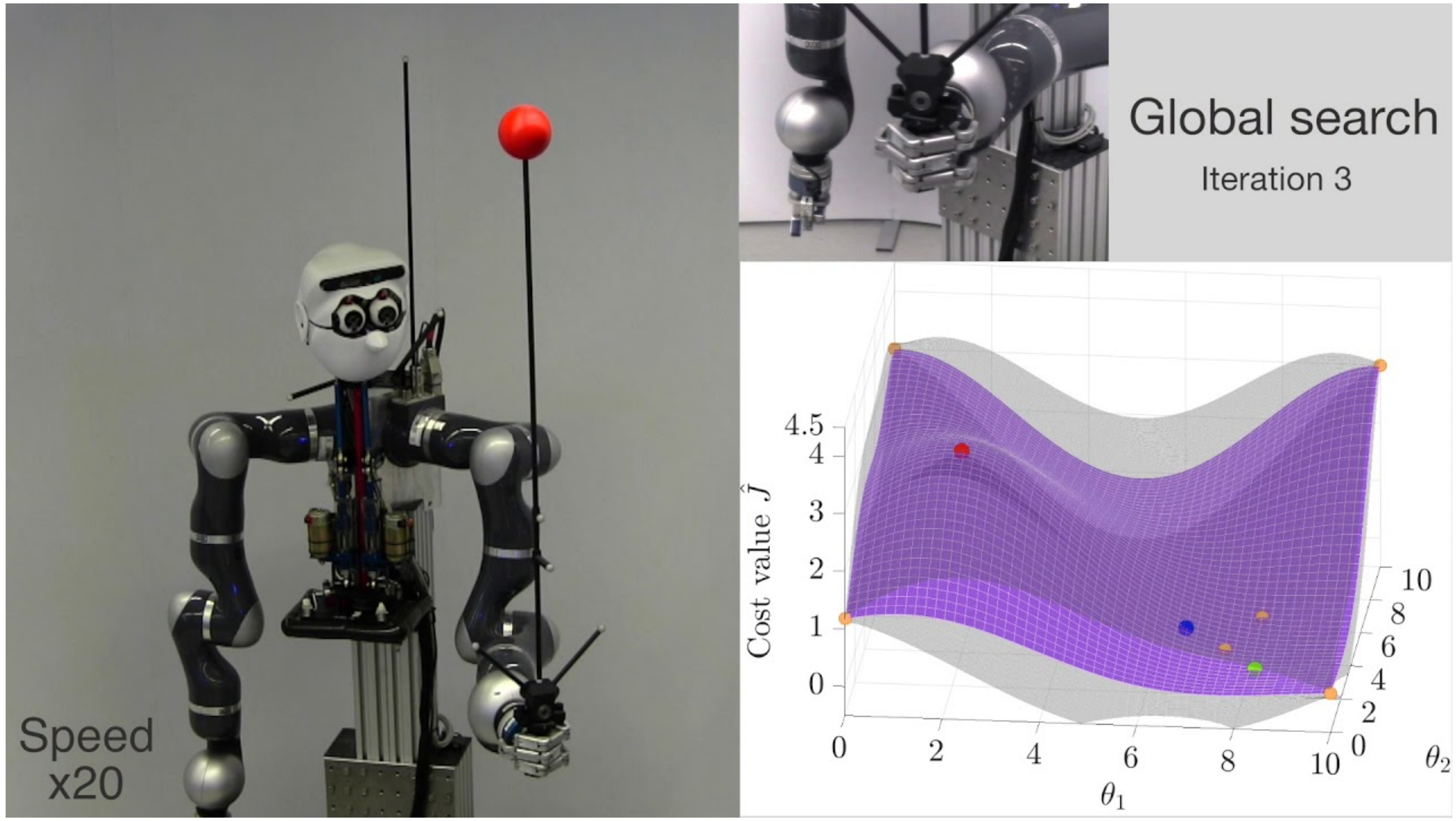
AutoML (Hyperparameter Optimization)

Automatic LQR Tuning Based on Gaussian Process Global Optimization

Sensor Placement Optimization



Golestan, Shadan, Omid Ardakanian, and Pierre Boulanger. "Grey-Box Bayesian Optimization for Sensor Placement in Assisted Living Environments." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. No. 20. 2024.



https://am.is.mpg.de/research_projects/cont-learn-bayes-opt

Reinforcement Learning

Reinforcement Learning

Introduction

- **Learning by interaction**
  - **Nature-inspired:** Learn how to achieve a particular task by trial and error

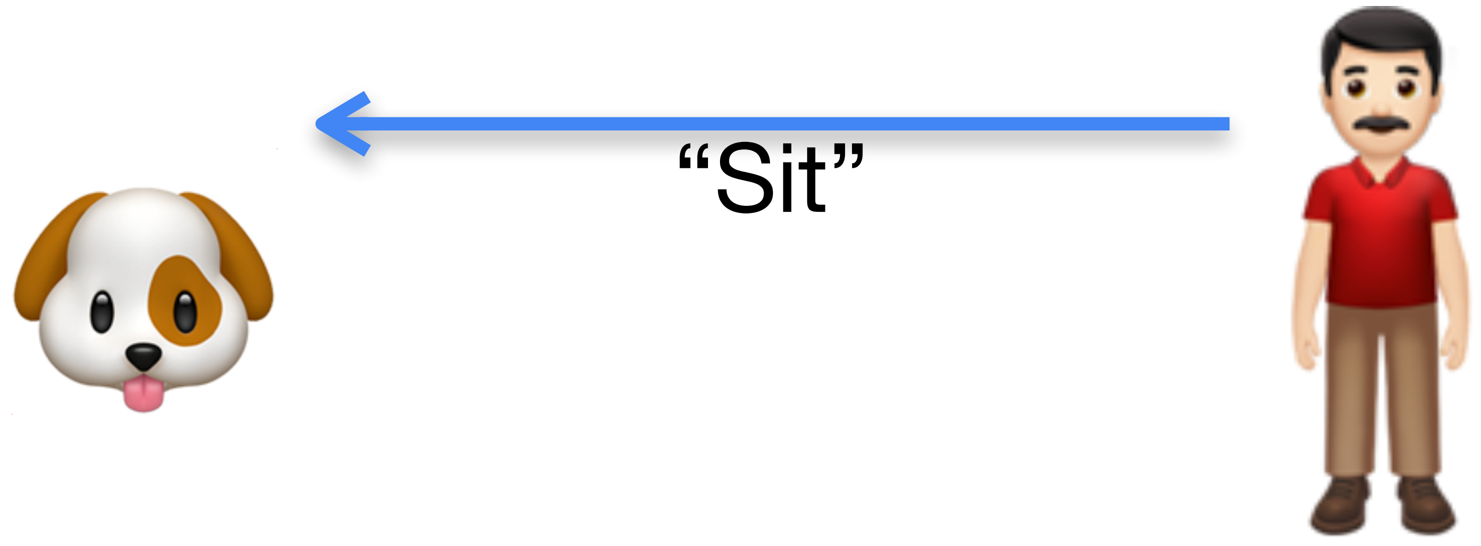
Reinforcement Learning
Introduction

- **Learning by interaction**
  - **Nature-inspired:** Learn how to achieve a particular task by trial and error

Reinforcement Learning

Introduction

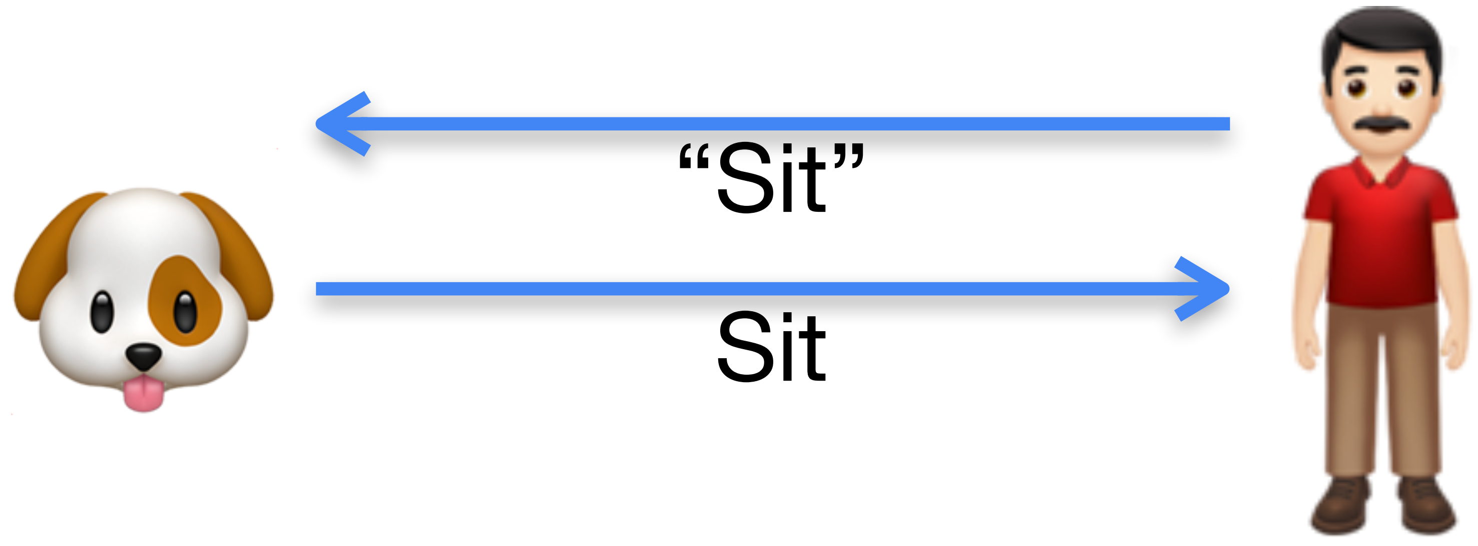
- **Learning by interaction**
  - **Nature-inspired:** Learn how to achieve a particular task by trial and error



Reinforcement Learning

Introduction

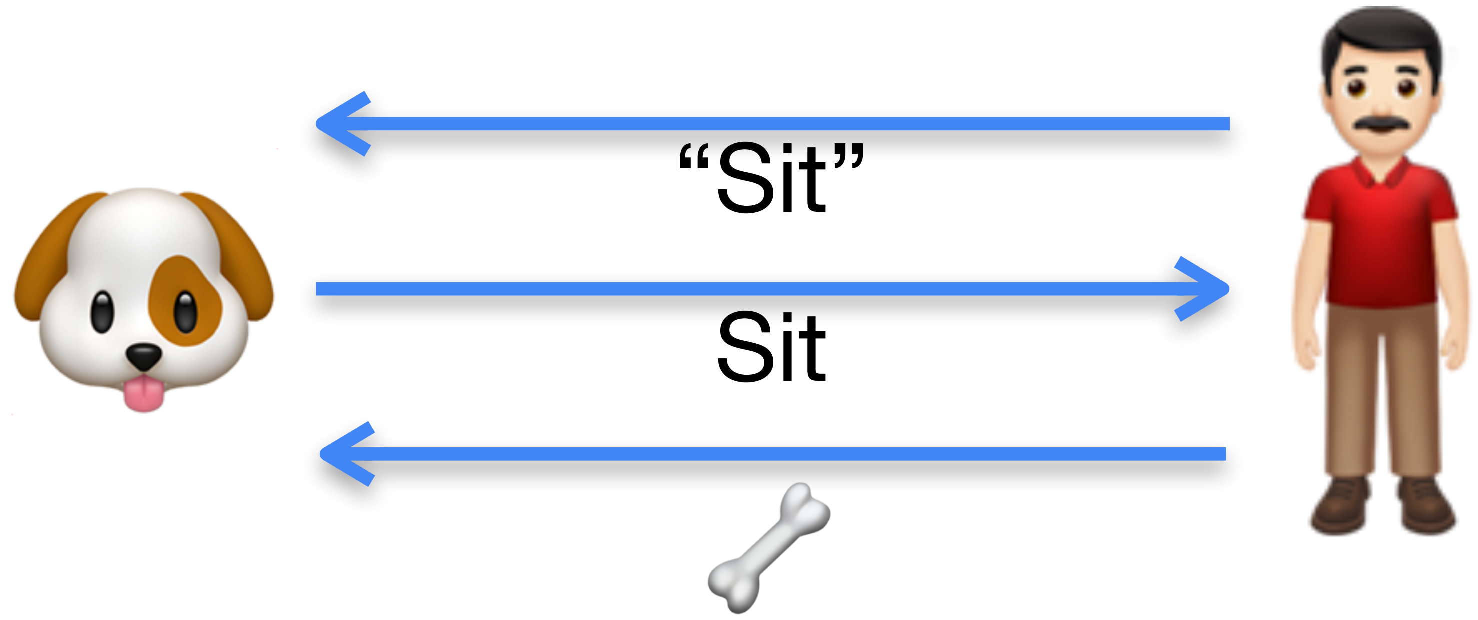
- **Learning by interaction**
  - **Nature-inspired:** Learn how to achieve a particular task by trial and error



Reinforcement Learning

Introduction

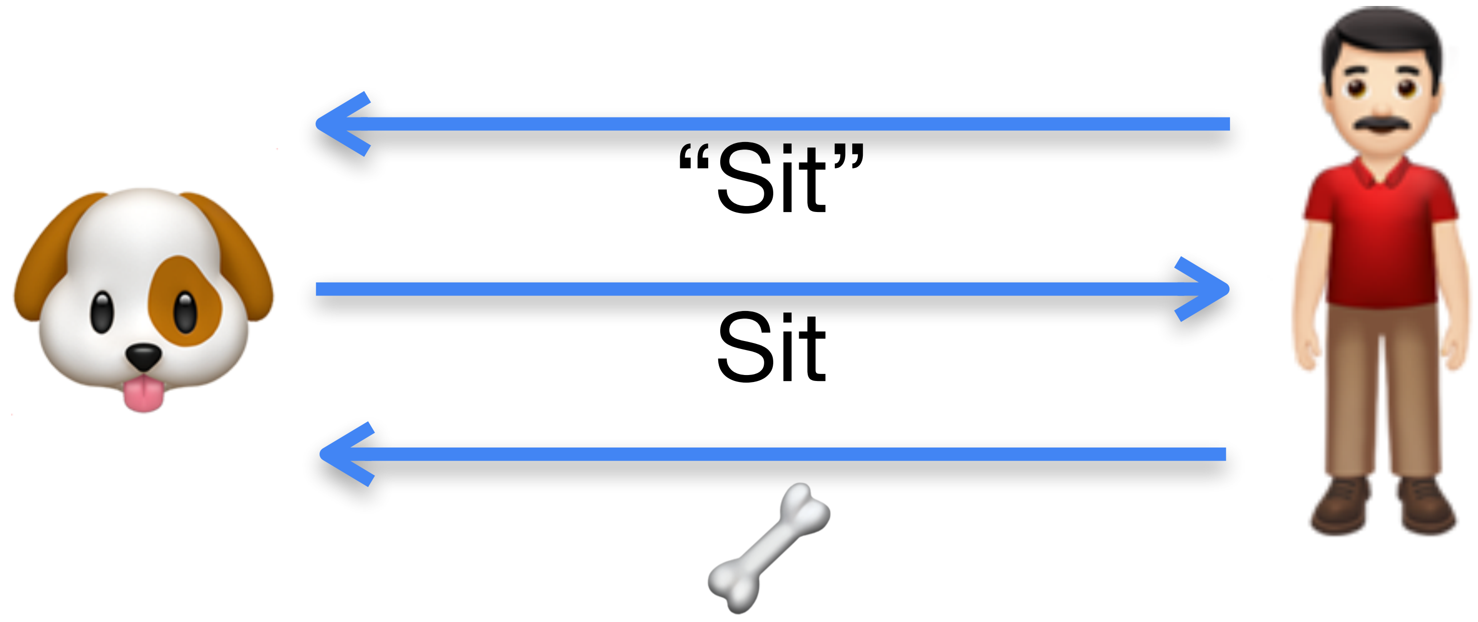
- **Learning by interaction**
  - **Nature-inspired:** Learn how to achieve a particular task by trial and error



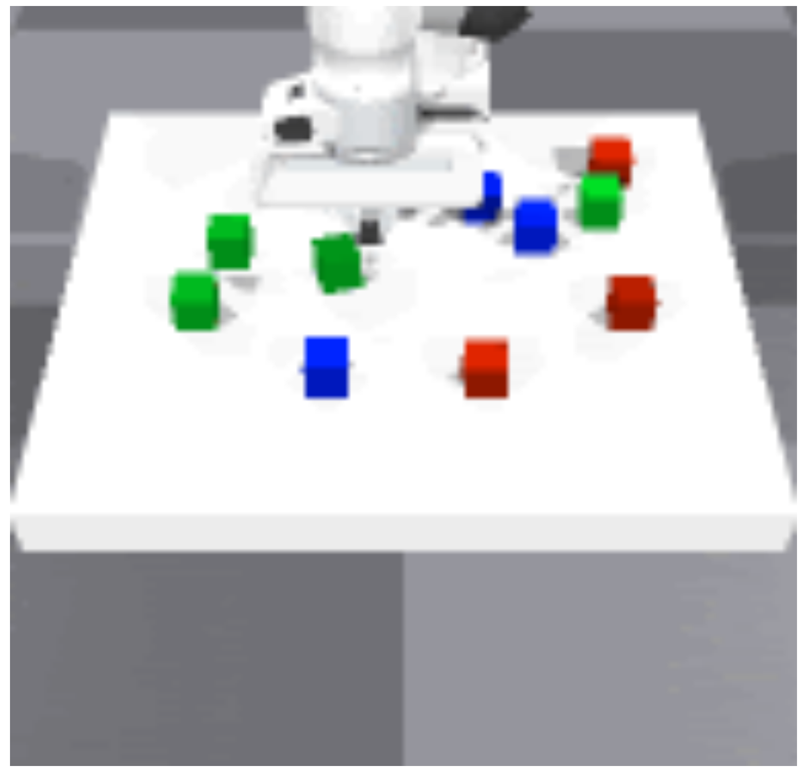
Reinforcement Learning

Introduction

- **Learning by interaction**
  - **Nature-inspired:** Learn how to achieve a particular task by trial and error



Object manipulation



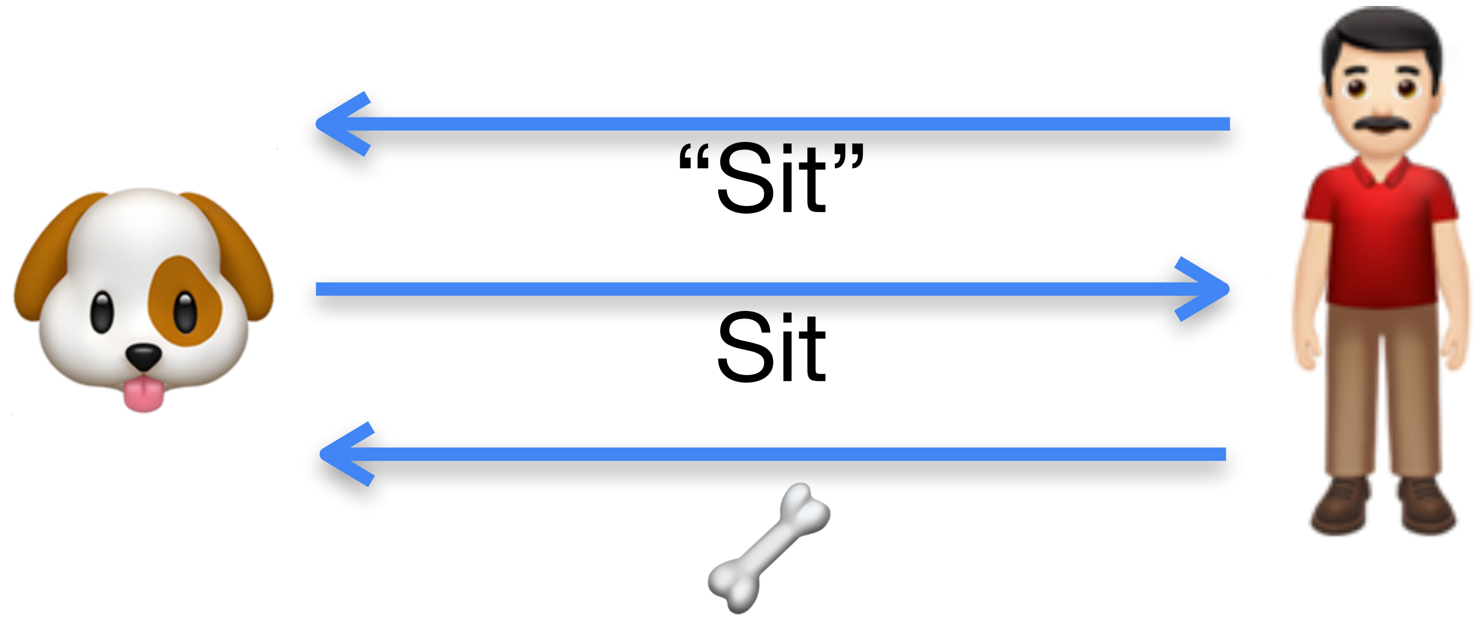
https://sites.google.com/view/entity-centric-rl

Haramati, Dan, Tal Daniel, and Aviv Tamar. "Entity-Centric Reinforcement Learning for Object Manipulation from Pixels." *arXiv preprint arXiv:2404.01220* (2024).

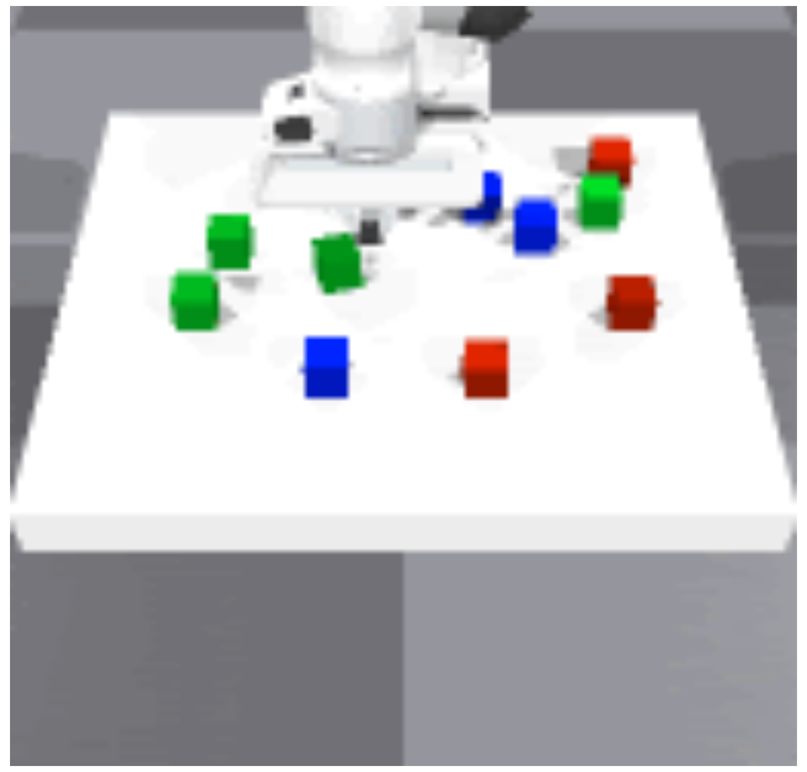
Reinforcement Learning

Introduction

- **Learning by interaction**
  - **Nature-inspired:** Learn how to achieve a particular task by trial and error



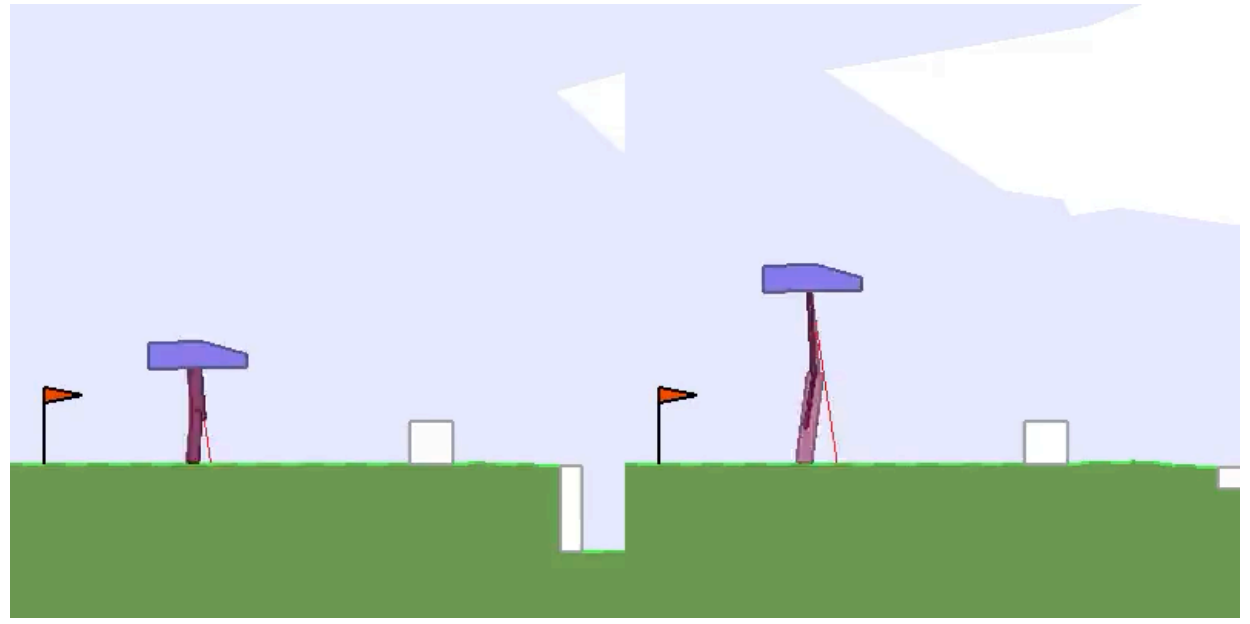
Object manipulation



https://sites.google.com/view/entity-centric-rl

Haramati, Dan, Tal Daniel, and Aviv Tamar. "Entity-Centric Reinforcement Learning for Object Manipulation from Pixels." *arXiv preprint arXiv:2404.01220* (2024).

Locomotion



Learning to navigate

Learns a body design jointly with the navigation task

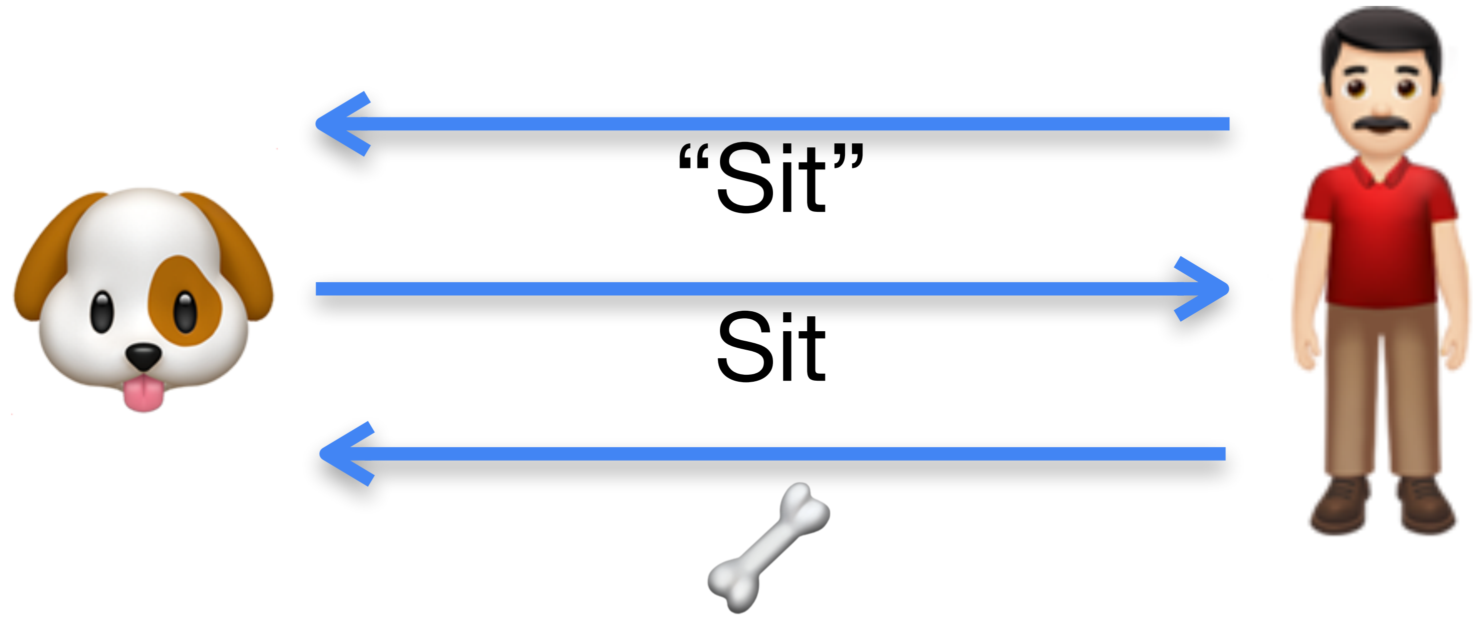
https://sites.google.com/view/entity-centric-rl

Ha, David. "Reinforcement learning for improving agent design." Artificial life 25.4 (2019): 352-365.

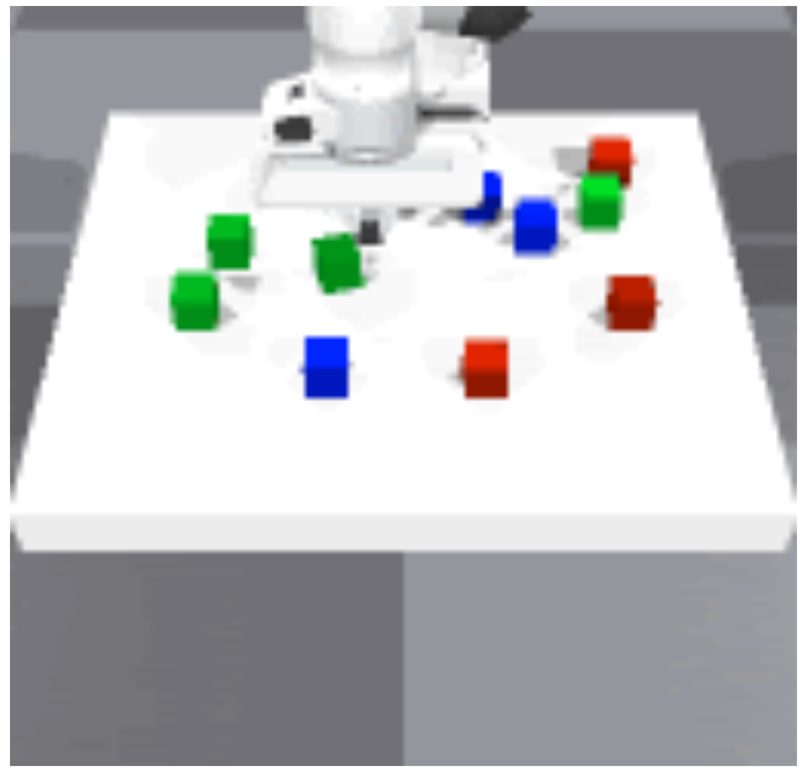
Reinforcement Learning

Introduction

- **Learning by interaction**
  - **Nature-inspired:** Learn how to achieve a particular task by trial and error



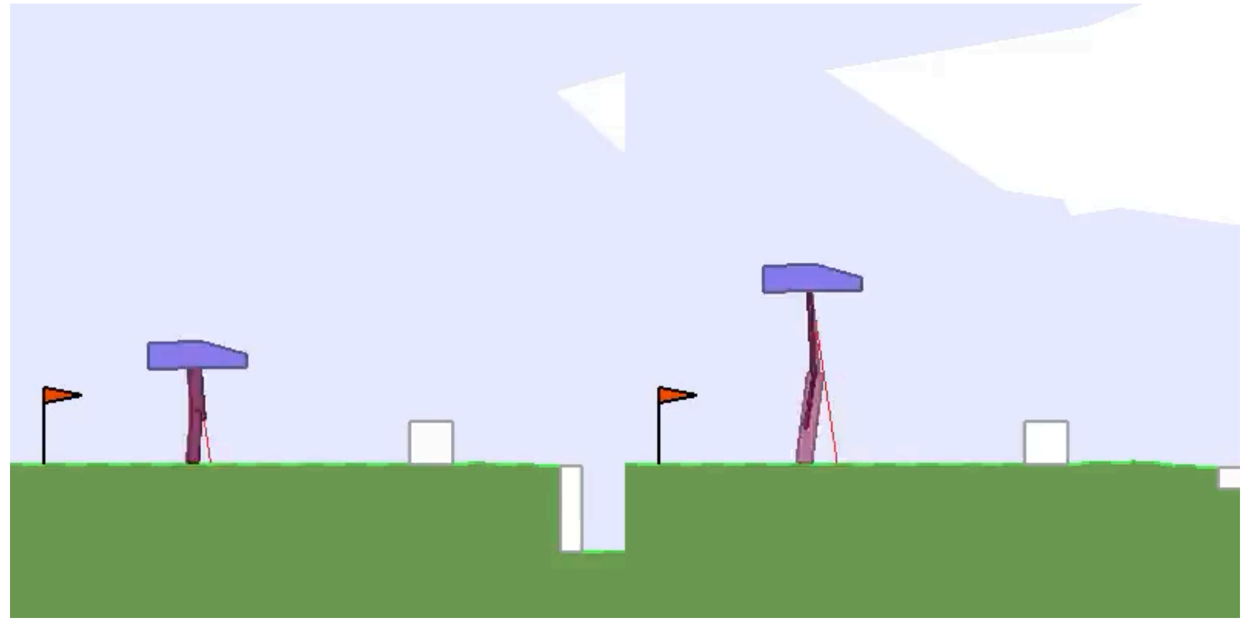
Object manipulation



https://sites.google.com/view/entity-centric-rl

Haramati, Dan, Tal Daniel, and Aviv Tamar. "Entity-Centric Reinforcement Learning for Object Manipulation from Pixels." *arXiv preprint arXiv:2404.01220* (2024).

Locomotion



Learning to navigate

Learns a body design jointly with the navigation task

https://sites.google.com/view/entity-centric-rl

Ha, David. "Reinforcement learning for improving agent design." Artificial life 25.4 (2019): 352-365.

Adaptation

https://anniesch.github.io/adapt-on-the-go/

Chen, Annie S., et al. "Adapt On-the-Go: Behavior Modulation for Single-Life Robot Deployment." arXiv preprint arXiv:2311.01059 (2023).

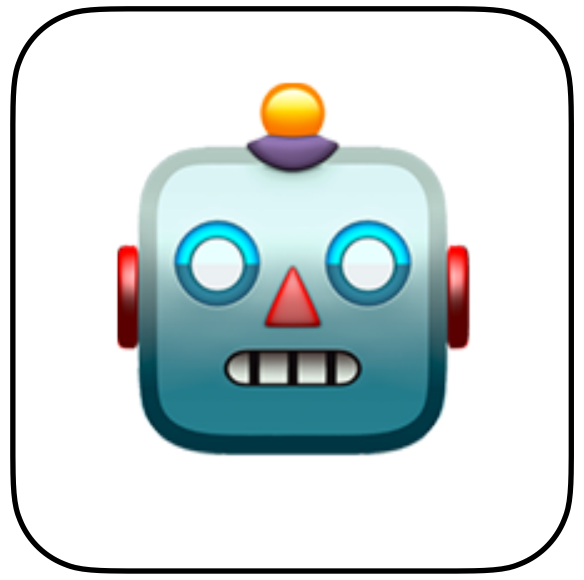
Reinforcement Learning

Core Concepts and Terminology

Reinforcement Learning

Core Concepts and Terminology

Agent:
the entity that
takes actions
e.g., robots

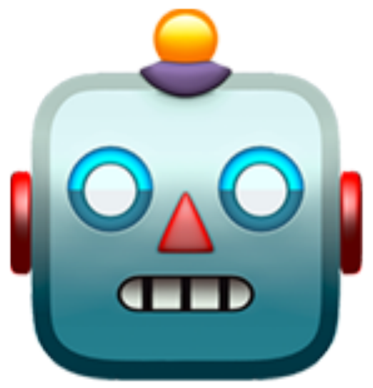


MITSUBISHI ELECTRIC UNIVERSITY OF ALBERTA amii

Reinforcement Learning

Core Concepts and Terminology

Agent:
the entity that
takes actions
e.g., robots

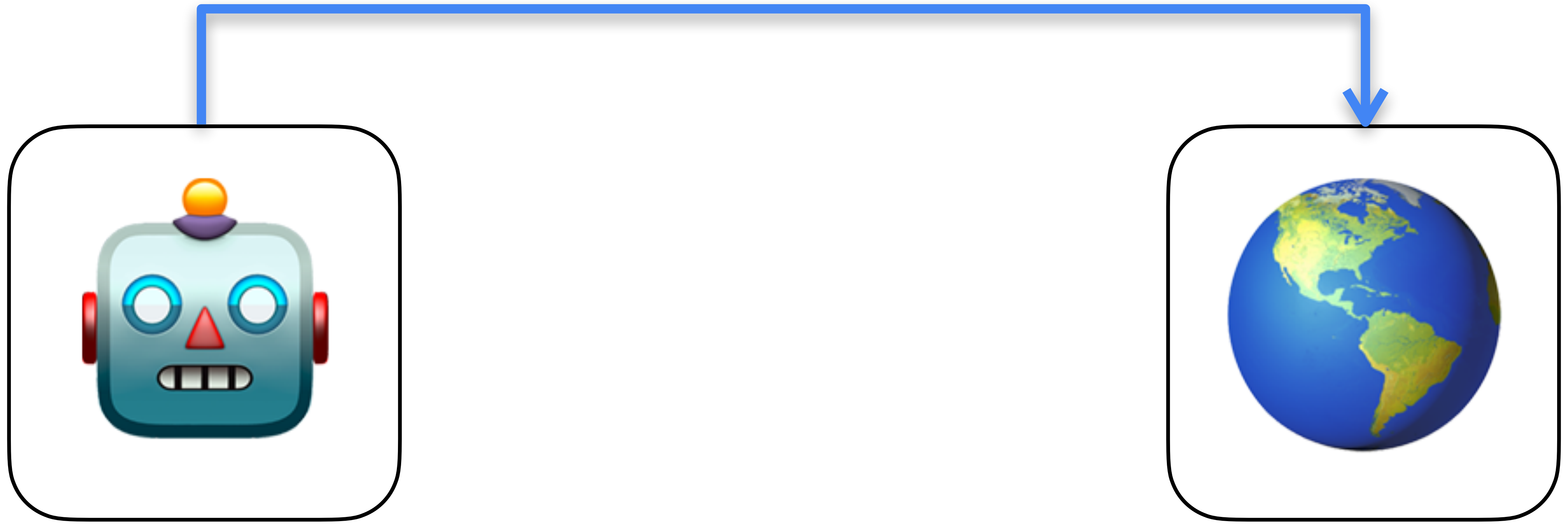


Environment:
the world where
the agent exists in
e.g., an assembly line

Reinforcement Learning

Core Concepts and Terminology

Actions (a_t):
the allowed moves
e.g. joints movements,
grab/drop objects



Agent:
the entity that
takes actions
e.g., robots

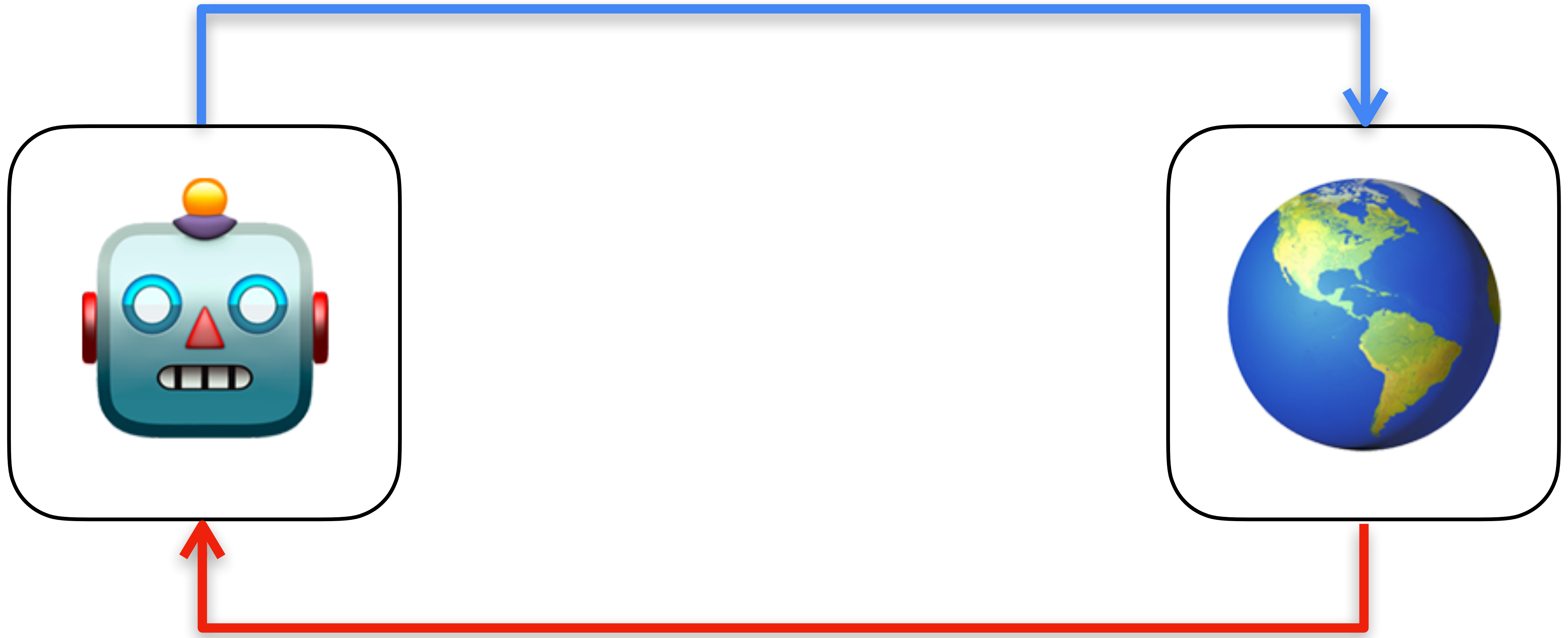
Environment:
the world where
the agent exists in
e.g., an assembly line

Reinforcement Learning

Core Concepts and Terminology

Actions (a_t):
the allowed moves
e.g. joints movements,
grab/drop objects

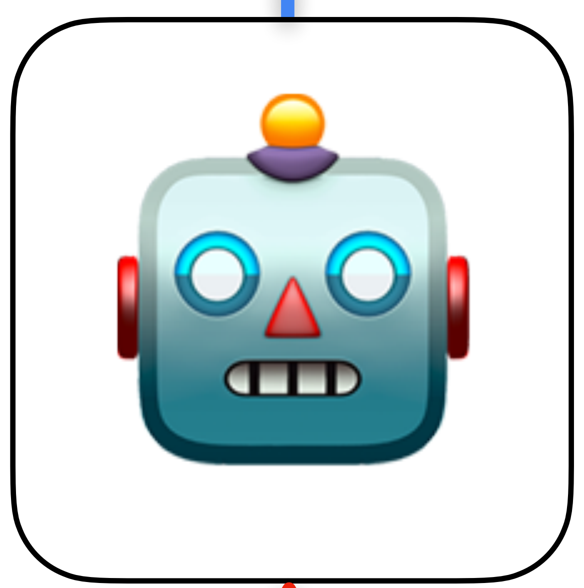
Agent:
the entity that
takes actions
e.g., robots



Environment:
the world where
the agent exists in
e.g., an assembly line

Observations:
changes of the environment
observable by the agent
e.g. objects locations

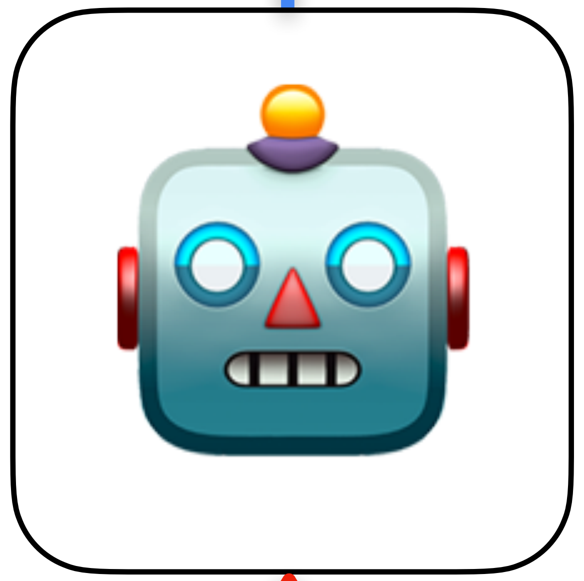
Agent:
the entity that
takes actions
e.g., robots



Environment:
the world where
the agent exists in
e.g., an assembly line

Observations:
changes of the environment
observable by the agent
e.g. objects locations

Agent:
the entity that
takes actions
e.g., robots

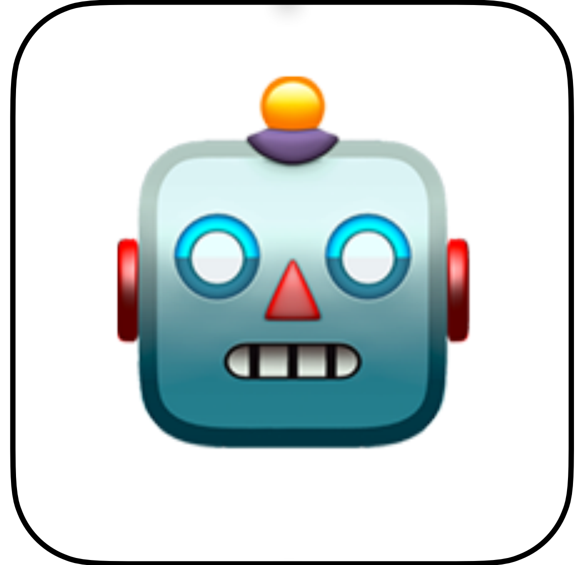


Environment:
the world where
the agent exists in
e.g., an assembly line

Observations:
changes of the environment
observable by the agent
e.g. objects locations

States (s_{t+1})
the immediate situation
the agent finds itself in

Agent:
the entity that
takes actions
e.g., robots



Environment:
the world where
the agent exists in
e.g., an assembly line

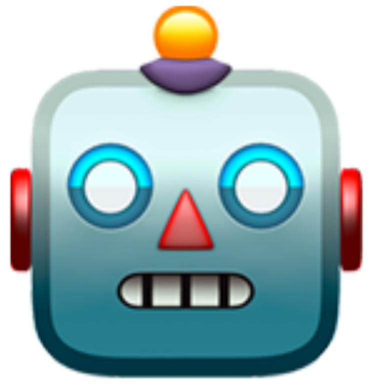
Observations:
changes of the environment
observable by the agent
e.g. objects locations

States (s_{t+1})
the immediate situation
the agent finds itself in

Reward (r_t)
a feedback to measure
the success/failure of a_t

MITSUBISHI ELECTRIC
UNIVERSITY OF ALBERTA
amii

Agent:
the entity that
takes actions
e.g., robots



Environment:
the world where
the agent exists in
e.g., an assembly line

Observations:
changes of the environment
observable by the agent
e.g. objects locations

States (s_{t+1})
the immediate situation
the agent finds itself in

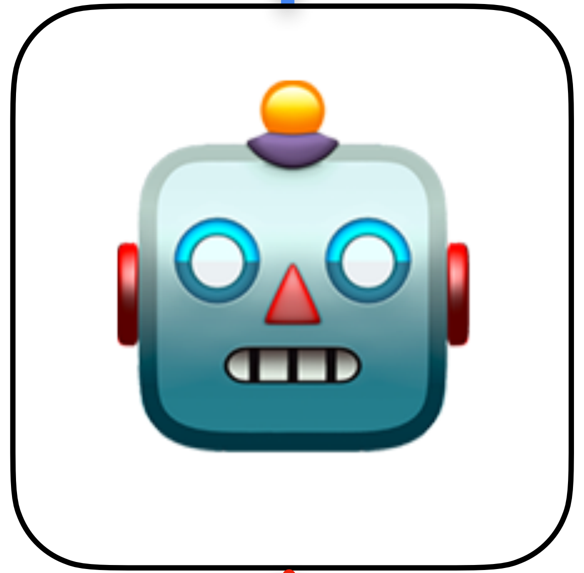
Reward (r_t)
a feedback to measure
the success/failure of a_t

Sum of Reward (R_t)

$$R_t = \sum_{i=t}^{H} r_t$$

H is a horizon (can be ∞)

Agent:
the entity that takes actions
e.g., robots



Environment:
the world where the agent exists in
e.g., an assembly line

Observations:
changes of the environment observable by the agent
e.g. objects locations

States (s_{t+1})
the immediate situation the agent finds itself in

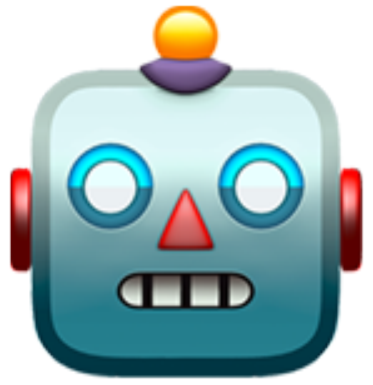
Reward (r_t)
a feedback to measure the success/failure of a_t

Sum of Reward (R_t)

$$R_t = \sum_{i=t}^{H} r_t = r_t + r_{t+1} + r_{t+2} \ldots + r_H$$

H is a horizon (can be ∞)

Agent:
the entity that takes actions
e.g., robots



Environment:
the world where the agent exists in
e.g., an assembly line

Observations:
changes of the environment observable by the agent
e.g. objects locations

States (s_{t+1})
the immediate situation the agent finds itself in

Reward (r_t)
a feedback to measure the success/failure of a_t

Sum of Reward (R_t)

$$R_t = \sum_{i=t}^{H} r_t = r_t + r_{t+1} + r_{t+2} \ldots + r_H$$

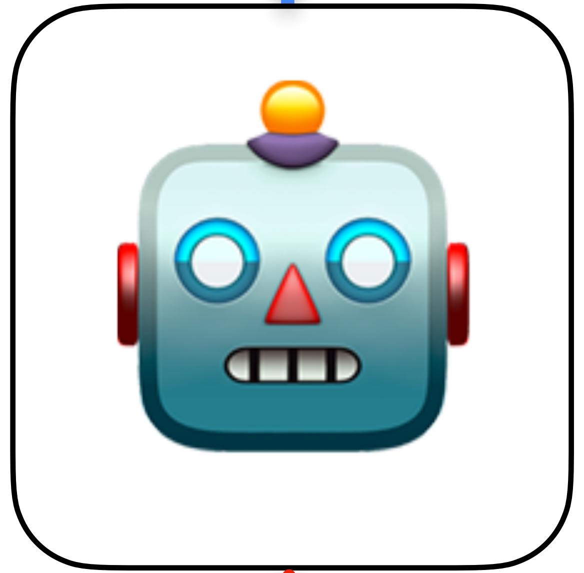
H **is a horizon (can be ∞)**

Discounted Sum of Reward (R_t)

$$R_t = \sum_{i=t}^{H} \gamma^i r_t$$

γ **is a discount factor ($0 < \gamma < 1$): makes future rewards worth less**

Agent:
the entity that
takes actions
e.g., robots



Environment:
the world where
the agent exists in
e.g., an assembly line

Observations:
changes of the environment
observable by the agent
e.g. objects locations

States (s_{t+1})
the immediate situation
the agent finds itself in

Reward (r_t)
a feedback to measure
the success/failure of a_t

Sum of Reward (R_t)

$$R_t = \sum_{i=t}^{H} r_t = r_t + r_{t+1} + r_{t+2} \ldots + r_H$$

H is a horizon (can be ∞)

Discounted Sum of Reward (R_t)

$$R_t = \sum_{i=t}^{H} \gamma^i r_t$$

γ is a discount factor ($0 < \gamma < 1$): makes future rewards worth less

💵 now OR 💵 1 year later

MITSUBISHI ELECTRIC UNIVERSITY OF ALBERTA amii

Agent:
the entity that
takes actions
e.g., robots

Environment:
the world where
the agent exists in
e.g., an assembly line

Observations:
changes of the environment
observable by the agent
e.g. objects locations

States (s_{t+1})
the immediate situation
the agent finds itself in

Reward (r_t)
a feedback to measure
the success/failure of a_t

Sum of Reward (R_t)

$$R_t = \sum_{i=t}^{H} r_t \; = r_t + r_{t+1} + r_{t+2} \ldots + r_H$$

H **is a horizon (can be ∞)**

Discounted Sum of Reward (R_t)

$$R_t = \sum_{i=t}^{H} \gamma^i r_t \; = \gamma^t r_t + \gamma^{t+1} r_{t+1} + \gamma^{t+2} r_{t+2} \ldots + \gamma^H r_H$$

γ **is a discount factor ($0 < \gamma < 1$): makes future rewards worth less**

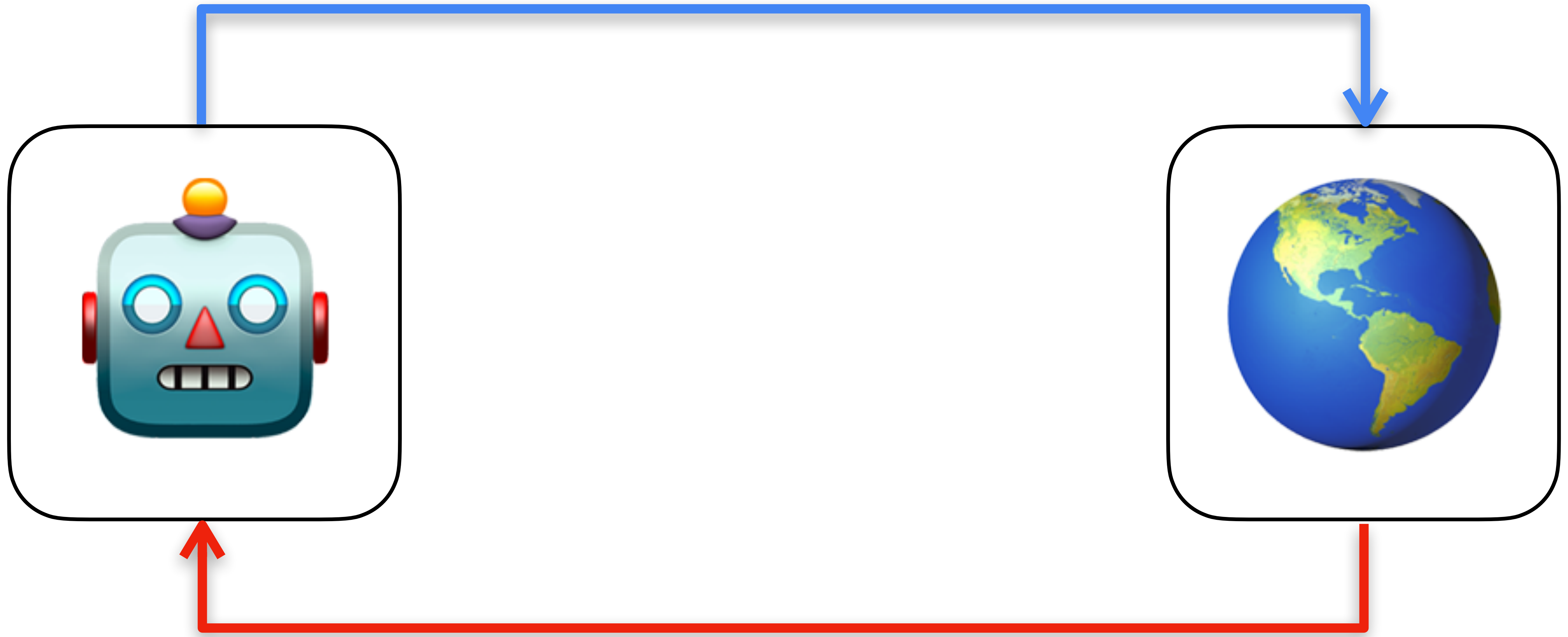
now **OR** **1 year later**

Reinforcement Learning

Core Concepts and Terminology

Actions (a_t):
the allowed moves
e.g. joints movements,
grab/drop objects

Agent:
the entity that
takes actions
e.g., robots



Environment:
the world where
the agent exists in
e.g., an assembly line

Observations:
changes of the environment
observable by the agent
e.g. objects locations

Reward (r_t) **States (s_{t+1})**

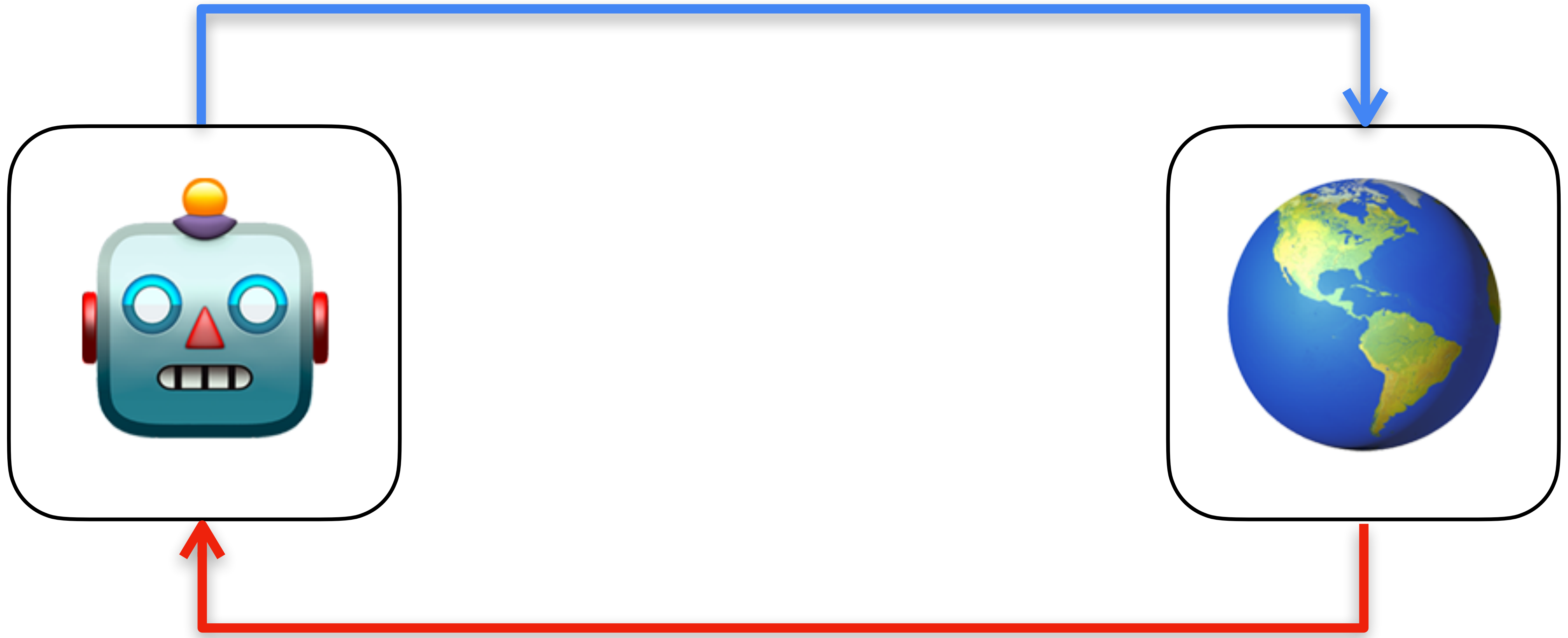
Discounted Sum of Reward (R_t)

Reinforcement Learning

Core Concepts and Terminology

Actions (a_t):
the allowed moves
e.g. joints movements,
grab/drop objects

Agent:
the entity that
takes actions
e.g., robots



Observations:
changes of the environment
observable by the agent
e.g. objects locations

Reward (r_t) **States (s_{t+1})**

Discounted Sum of Reward (R_t)

Reinforcement Learning

Core Concepts and Terminology

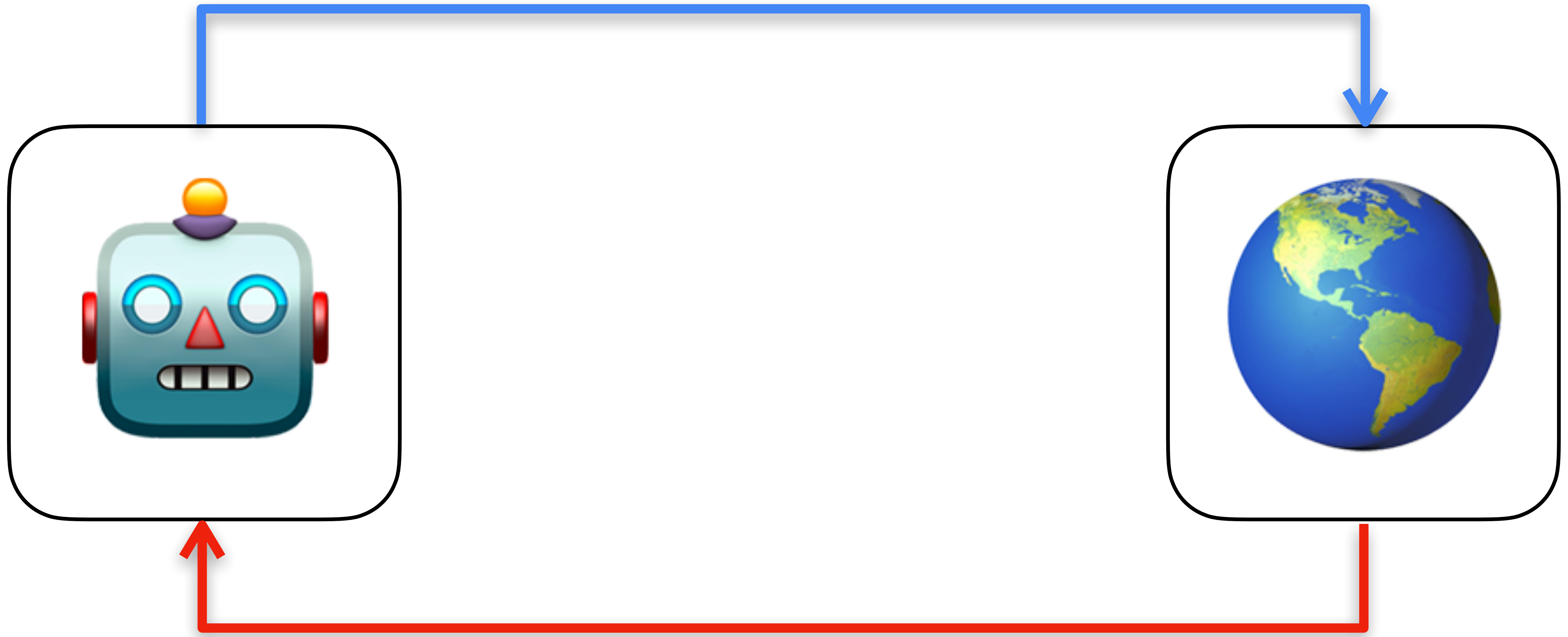
Q-function

$$Q(s_t, a_t)$$

expected total future reward that the agent can receive in state s_t by taking action a_t

Actions (a_t):
the allowed moves
e.g. joints movements, grab/drop objects

Agent:
the entity that takes actions
e.g., robots



Observations:
changes of the environment observable by the agent
e.g. objects locations

Reward (r_t) **States (s_{t+1})**

Discounted Sum of Reward (R_t)

Reinforcement Learning

Core Concepts and Terminology

Q-function

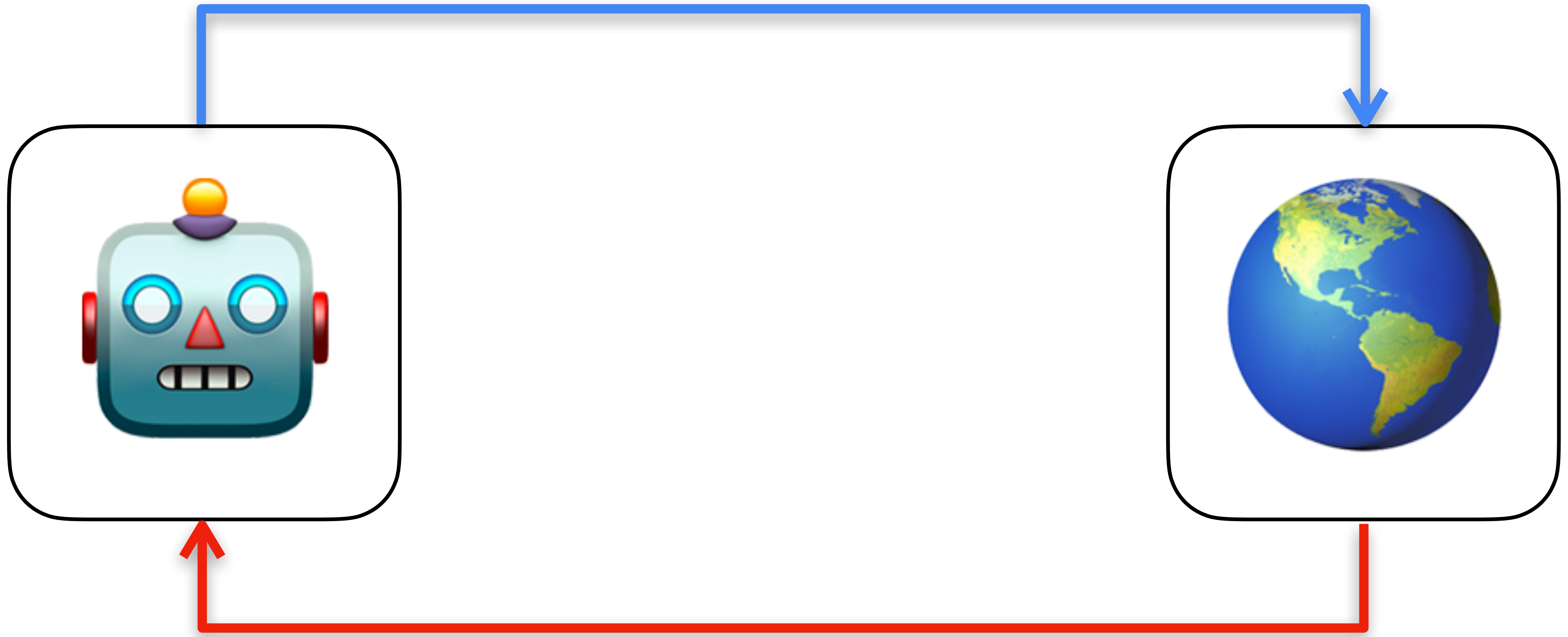
$Q(s_t, a_t)$

expected total future reward that the agent can receive in state s_t by taking action a_t

$$Q(s_t, a_t) = E[R_t \,|\, s_t, a_t]$$

Actions (a_t):
the allowed moves
e.g. joints movements, grab/drop objects

Agent:
the entity that takes actions
e.g., robots



Observations:
changes of the environment observable by the agent
e.g. objects locations

Reward (r_t) **States (s_{t+1})**

Discounted Sum of Reward (R_t)

MITSUBISHI ELECTRIC UNIVERSITY OF ALBERTA amii

Reinforcement Learning

Core Concepts and Terminology

Q-function

$$Q(s_t, a_t)$$

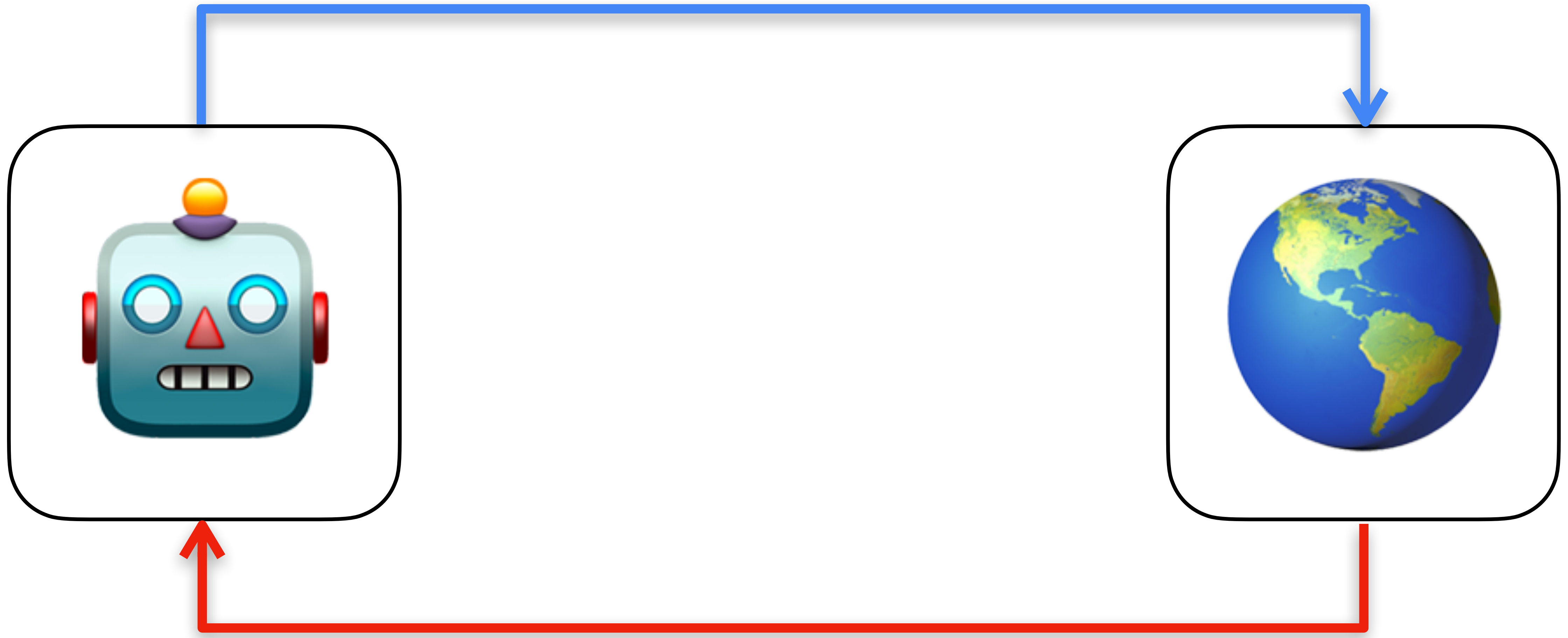
expected total future reward that the agent can receive in state s_t by taking action a_t

$$Q(s_t, a_t) = E[R_t \mid s_t, a_t]$$

If the agent "magically" knows $Q(s_t, a_t)$, how it should take action a_t at any given state?

Actions (a_t):
the allowed moves
e.g. joints movements, grab/drop objects

Agent:
the entity that takes actions
e.g., robots



Observations:
changes of the environment observable by the agent
e.g. objects locations

Reward (r_t) **States (s_{t+1})**

Discounted Sum of Reward (R_t)

Reinforcement Learning

Core Concepts and Terminology

Q-function

$$Q(s_t, a_t)$$

expected total future reward that the agent can receive in state s_t by taking action a_t

$$Q(s_t, a_t) = E[R_t \mid s_t, a_t]$$

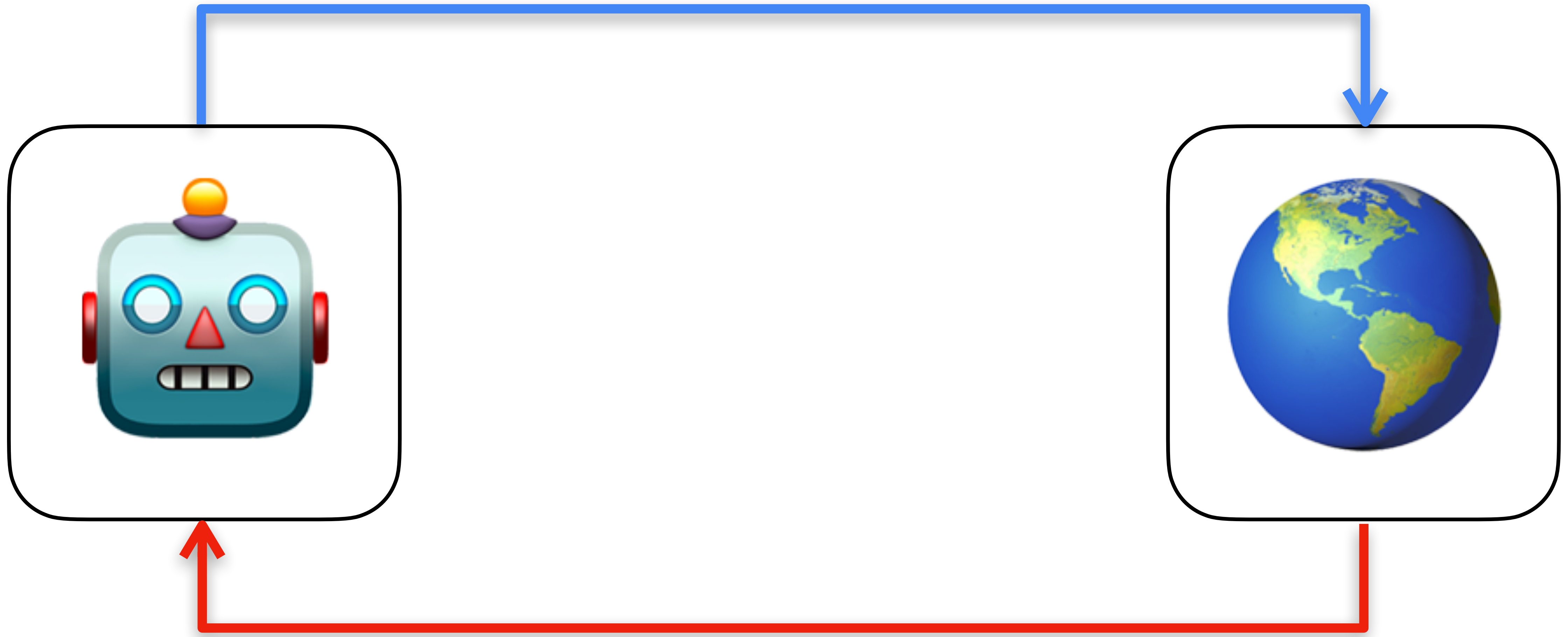
If the agent "magically" knows $Q(s_t, a_t)$, how it should take action a_t at any given state?

Policy π

best action to take at any given state s

Actions (a_t):
the allowed moves
e.g. joints movements, grab/drop objects

Agent:
the entity that takes actions
e.g., robots



Observations:
changes of the environment observable by the agent
e.g. objects locations

Reward (r_t) **States (s_{t+1})**

Discounted Sum of Reward (R_t)

Reinforcement Learning

Core Concepts and Terminology

Q-function

$$Q(s_t, a_t)$$

expected total future reward that the agent can receive in state s_t by taking action a_t

$$Q(s_t, a_t) = E[R_t \mid s_t, a_t]$$

If the agent "magically" knows $Q(s_t, a_t)$, how it should take action a_t at any given state?

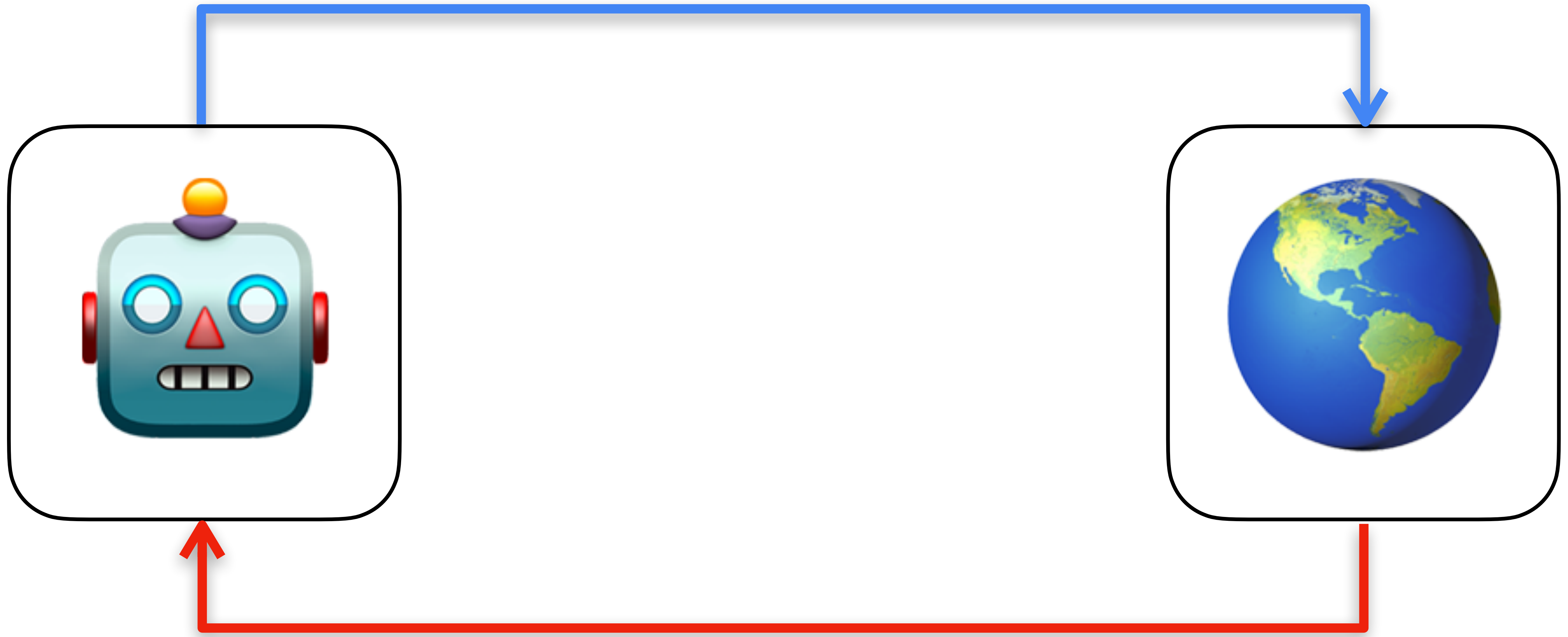
Policy π

best action to take at any given state s

$$\pi^*(s) = \underset{a}{\arg\max}\, Q(s, a)$$

Actions (a_t):
the allowed moves
e.g. joints movements, grab/drop objects

Agent:
the entity that takes actions
e.g., robots



Observations:
changes of the environment observable by the agent
e.g. objects locations

Reward (r_t) **States (s_{t+1})**

Discounted Sum of Reward (R_t)

Reinforcement Learning

Core Concepts and Terminology

Action

states

	a_0	a_1	a_2	...
s_0	$Q(s_0, a_0)$	$Q(s_0, a_1)$	$Q(s_0, a_2)$	...
s_1	$Q(s_1, a_0)$	$Q(s_1, a_1)$	$Q(s_1, a_2)$	...
s_2	$Q(s_2, a_0)$	$Q(s_2, a_1)$	$Q(s_2, a_2)$	...
...	...	...	...	...

Q-function

$Q(s_t, a_t)$

expected total future reward that the agent can receive in state s_t by taking action a_t

$$Q(s_t, a_t) = E[R_t \mid s_t, a_t]$$

🧐 If the agent "magically" knows $Q(s_t, a_t)$, how it should take action a_t at any given state?

Policy π

best action to take at any given state s

$$\pi^*(s) = \arg\max_a Q(s, a)$$

Reinforcement Learning

Core Concepts and Terminology

Q-function

$Q(s_t, a_t)$

expected total future reward that the agent can receive in state s_t by taking action a_t

$$Q(s_t, a_t) = E[R_t \mid s_t, a_t]$$

If the agent “magically” knows $Q(s_t, a_t)$, how it should take action a_t at any given state?

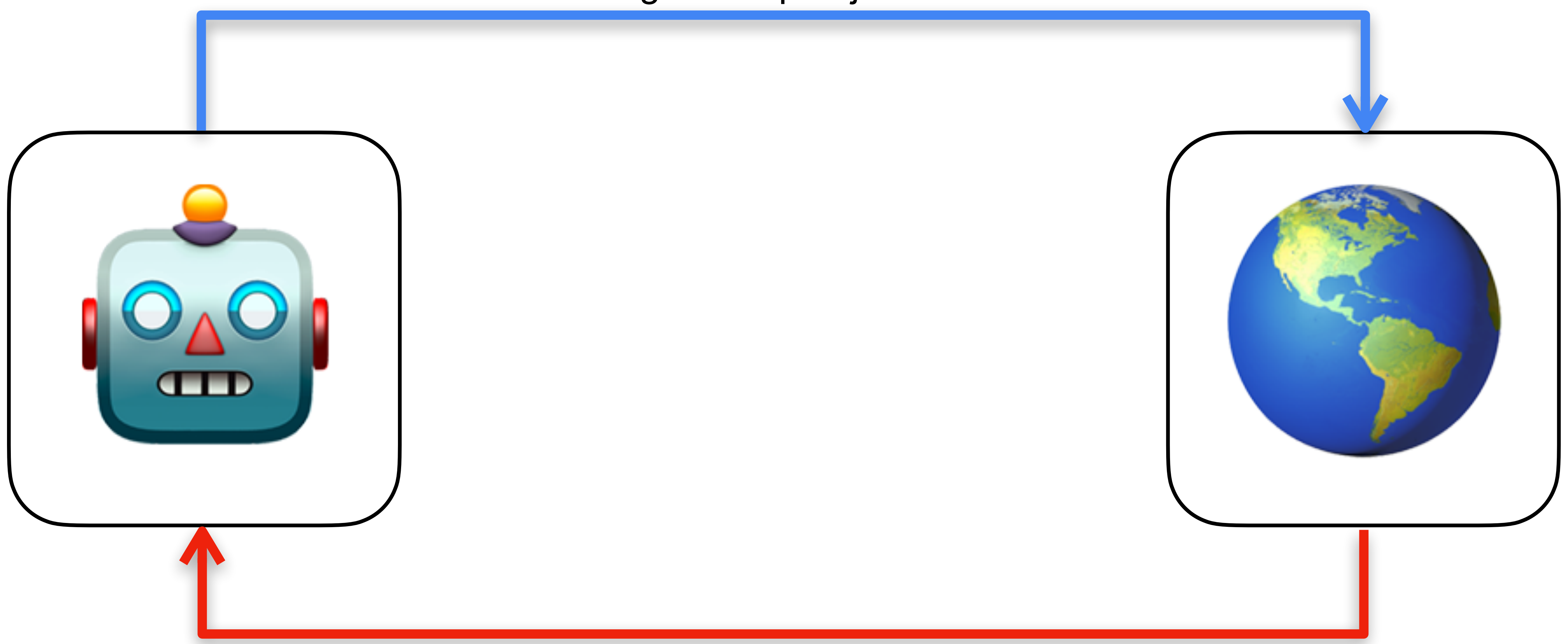
Policy π

best action to take at any given state s

$$\pi^*(s) = \arg\max_a Q(s, a)$$

Actions (a_t):
the allowed moves
e.g. joints movements, grab/drop objects

Agent:
the entity that takes actions
e.g., robots



Observations:
changes of the environment observable by the agent
e.g. objects locations

Reward (r_t) **States (s_{t+1})**

Discounted Sum of Reward (R_t)

Reinforcement Learning

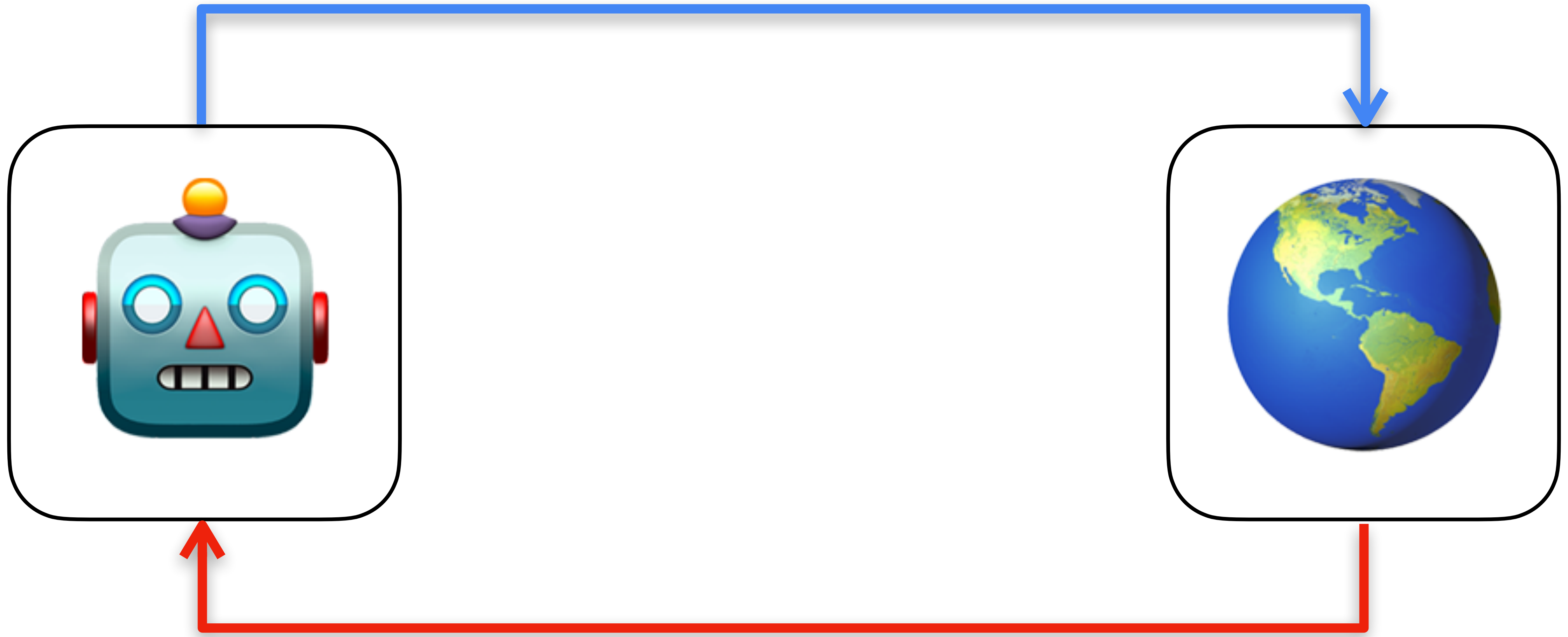
Core Concepts and Terminology

Actions (a_t):
the allowed moves
e.g. joints movements, grab/drop objects

Q-function
$Q(s_t, a_t)$

Policy π

Agent:
the entity that takes actions
e.g., robots



Environment:
the world where the agent exists in
e.g., an assembly line

Observations:
changes of the environment observable by the agent
e.g. objects locations

Reward (r_t) **States (s_{t+1})**

Discounted Sum of Reward (R_t)

Reinforcement Learning

Core Concepts and Terminology

Q-function

$Q(s_t, a_t)$

Policy π

Actions (a_t):

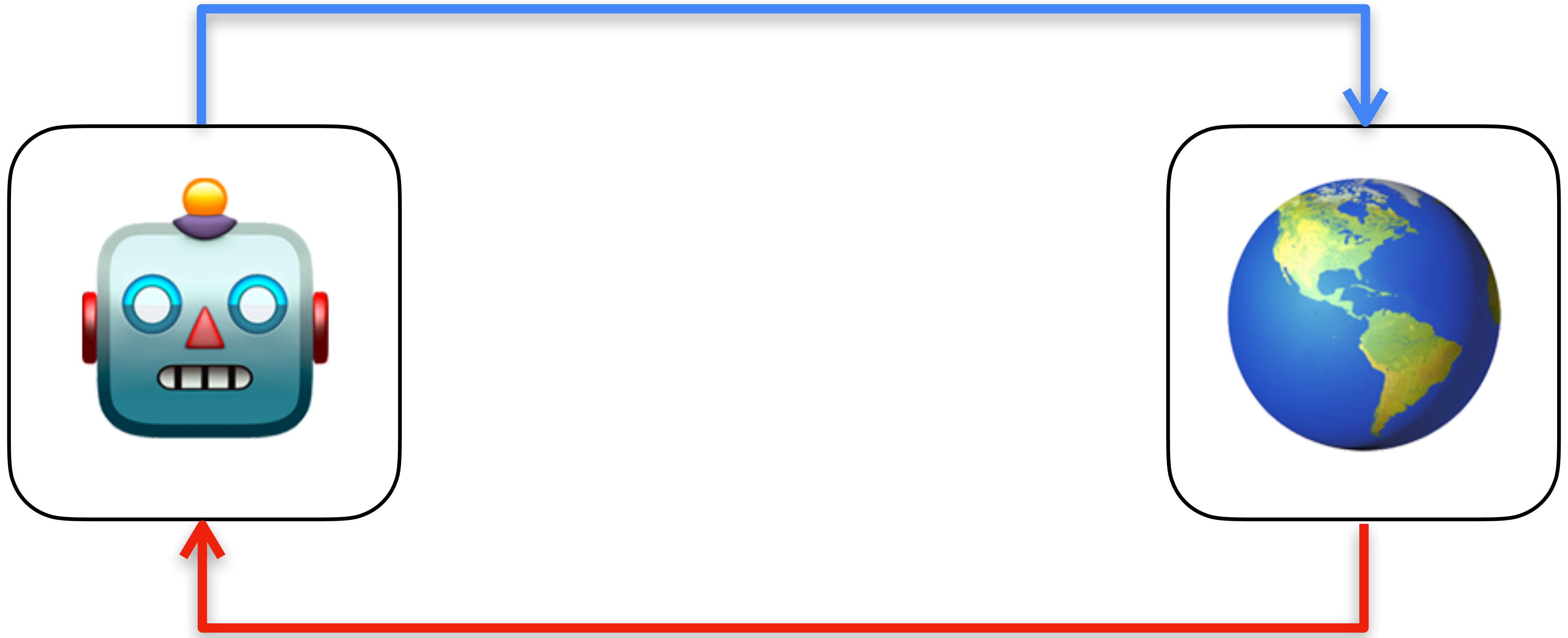
the allowed moves

e.g. joints movements, grab/drop objects

Agent:

the entity that takes actions

e.g., robots



Environment:

the world where the agent exists in

e.g., an assembly line

Observations:

changes of the environment observable by the agent

e.g. objects locations

Reward (r_t) **States (s_{t+1})**

Discounted Sum of Reward (R_t)

Reinforcement Learning

Q-Learning

Reinforcement Learning

Q-Learning

1. Initialize π

Reinforcement Learning

Q-Learning

1. Initialize π
2. $t = 0$

Reinforcement Learning

Q-Learning

1. Initialize π
2. $t = 0$
3. Initialize $s_t = s_0$

Reinforcement Learning

Q-Learning

1. Initialize π
2. $t = 0$
3. Initialize $s_t = s_0$
4. Loop:

Reinforcement Learning

Q-Learning

1. Initialize π
2. $t = 0$
3. Initialize $s_t = s_0$
4. Loop:
    4.1. Take $a_t = \pi(s_t)$

Reinforcement Learning

Q-Learning

1. Initialize π
2. $t = 0$
3. Initialize $s_t = s_0$
4. Loop:
   4.1. Take $a_t = \pi(s_t)$
   4.2. Observe (r_t, s_{t+1})

Reinforcement Learning

Q-Learning

1. Initialize π
2. $t = 0$
3. Initialize $s_t = s_0$
4. Loop:
    4.1. Take $a_t = \pi(s_t)$
    4.2. Observe (r_t, s_{t+1})
    4.3. $Q(s_t, a_t) =$

Reinforcement Learning

Q-Learning

1. Initialize π
2. $t = 0$
3. Initialize $s_t = s_0$
4. Loop:
    4.1. Take $a_t = \pi(s_t)$
    4.2. Observe (r_t, s_{t+1})
    4.3. $Q(s_t, a_t) = (1 - \alpha) \qquad\qquad + \ \alpha(\qquad\qquad)$

Reinforcement Learning

Q-Learning

1. Initialize π
2. $t = 0$
3. Initialize $s_t = s_0$
4. Loop:
   4.1. Take $a_t = \pi(s_t)$
   4.2. Observe (r_t, s_{t+1})
   4.3. $Q(s_t, a_t) = (1 - \alpha)\, Q(s_t, a_t) \;+\; \alpha(\qquad\qquad\qquad)$

Reinforcement Learning

Q-Learning

1. Initialize π
2. $t = 0$
3. Initialize $s_t = s_0$
4. Loop:
    4.1. Take $a_t = \pi(s_t)$
    4.2. Observe (r_t, s_{t+1})
    4.3. $Q(s_t, a_t) = (1 - \alpha)\, Q(s_t, a_t) \;+\; \alpha(\, r_t + \gamma \max_{a'} Q(s_{t+1}, a')\,)$

Reinforcement Learning

Q-Learning

1. Initialize π
2. $t = 0$
3. Initialize $s_t = s_0$
4. Loop:
   4.1. Take $a_t = \pi(s_t)$
   4.2. Observe (r_t, s_{t+1})
   4.3. $Q(s_t, a_t) = (1 - \alpha)\, Q(s_t, a_t) \;+\; \alpha(\, r_t + \gamma \max_{a'} Q(s_{t+1}, a')\,)$
   4.4 With prob. $1 - \epsilon$: $\pi(s_t) = \arg\max_{a} Q(s_t, a)$; else choose a random action

Reinforcement Learning

Q-Learning

1. Initialize π
2. $t = 0$
3. Initialize $s_t = s_0$
4. Loop:
    - 4.1. Take $a_t = \pi(s_t)$
    - 4.2. Observe (r_t, s_{t+1})
    - 4.3. $Q(s_t, a_t) = (1 - \alpha)\, Q(s_t, a_t) \;+\; \alpha(\, r_t + \gamma \max_{a'} Q(s_{t+1}, a')\,)$
    - 4.4 With prob. $1 - \epsilon$: $\pi(s_t) = \arg\max_a Q(s_t, a)$; else choose a random action
    - 4.5. $t = t + 1$

Reinforcement Learning

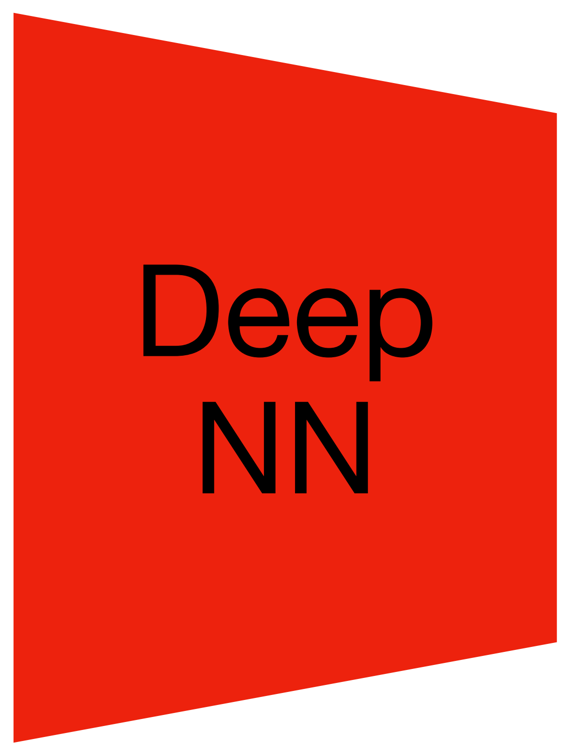
Deep Q Network (DQN)

Main Idea: use neural networks to model Q-Function

Reinforcement Learning

Deep Q Network (DQN)

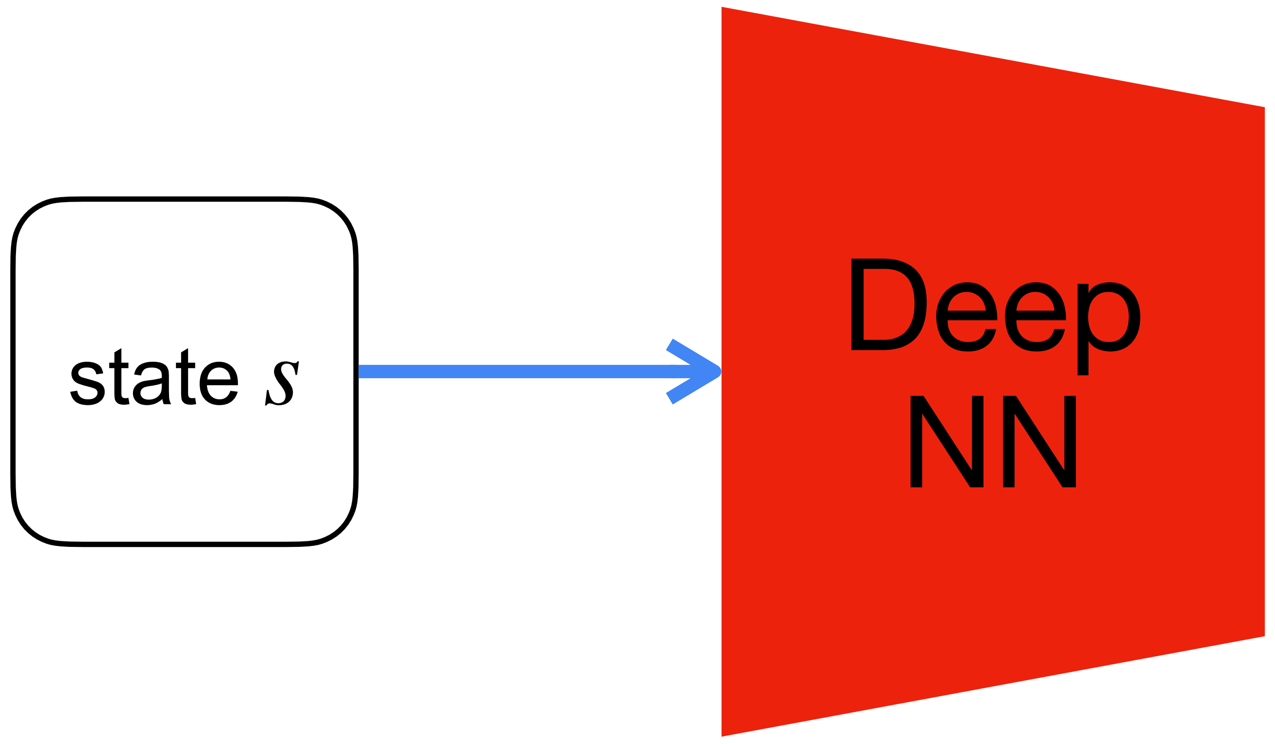
Main Idea: use neural networks to model Q-Function



Reinforcement Learning

Deep Q Network (DQN)

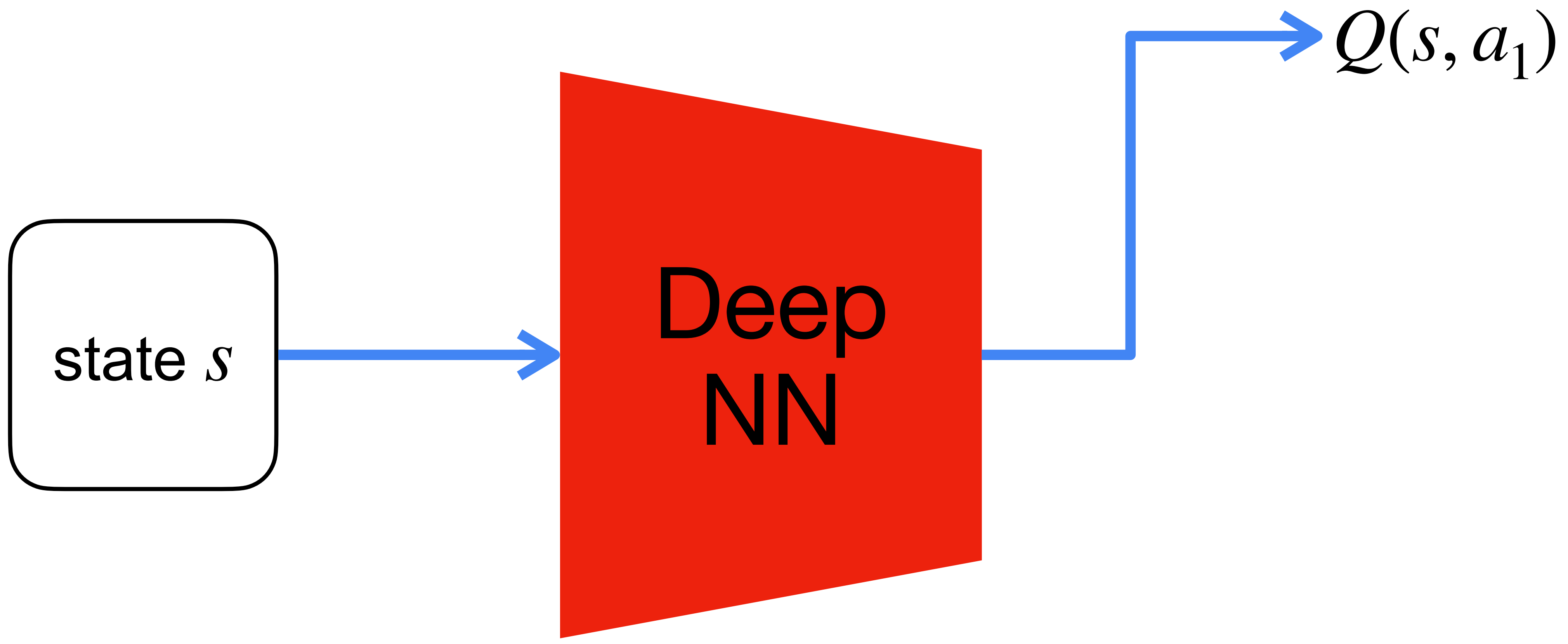
Main Idea: use neural networks to model Q-Function



Reinforcement Learning

Deep Q Network (DQN)

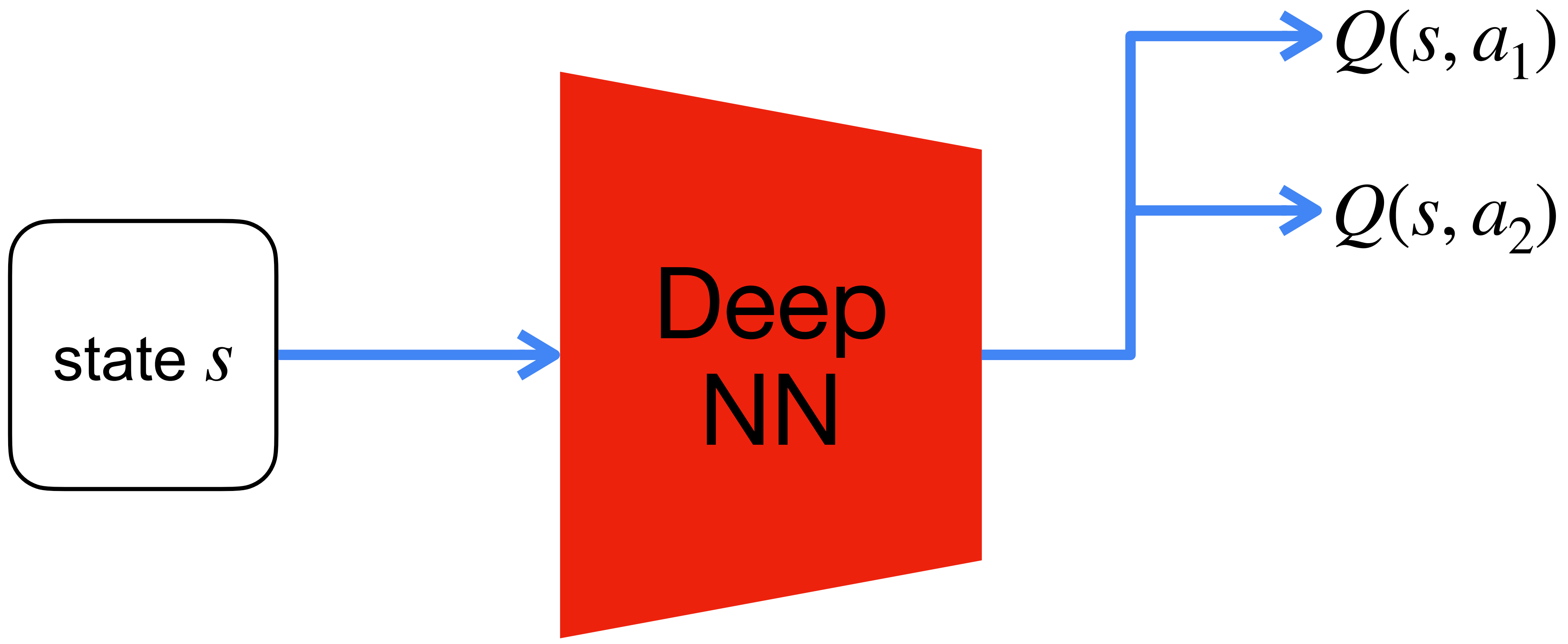
Main Idea: use neural networks to model Q-Function



Reinforcement Learning

Deep Q Network (DQN)

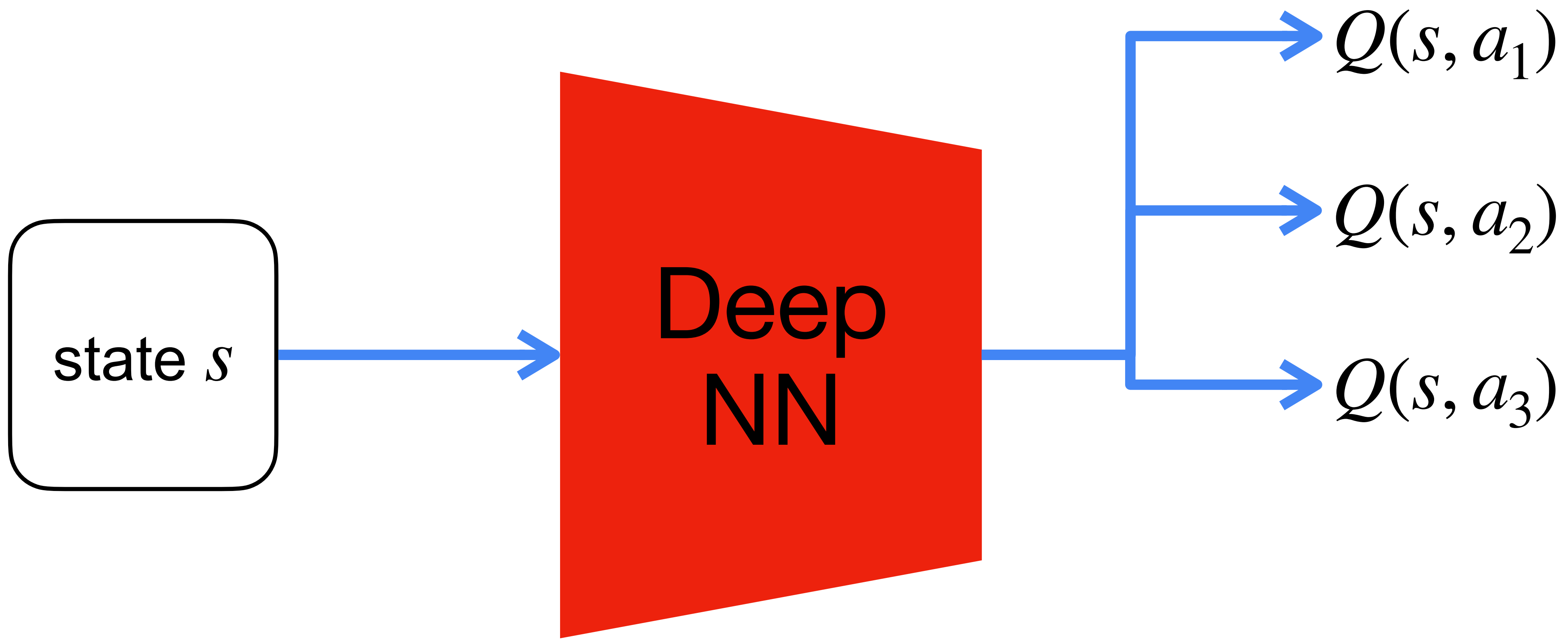
Main Idea: use neural networks to model Q-Function



Reinforcement Learning

Deep Q Network (DQN)

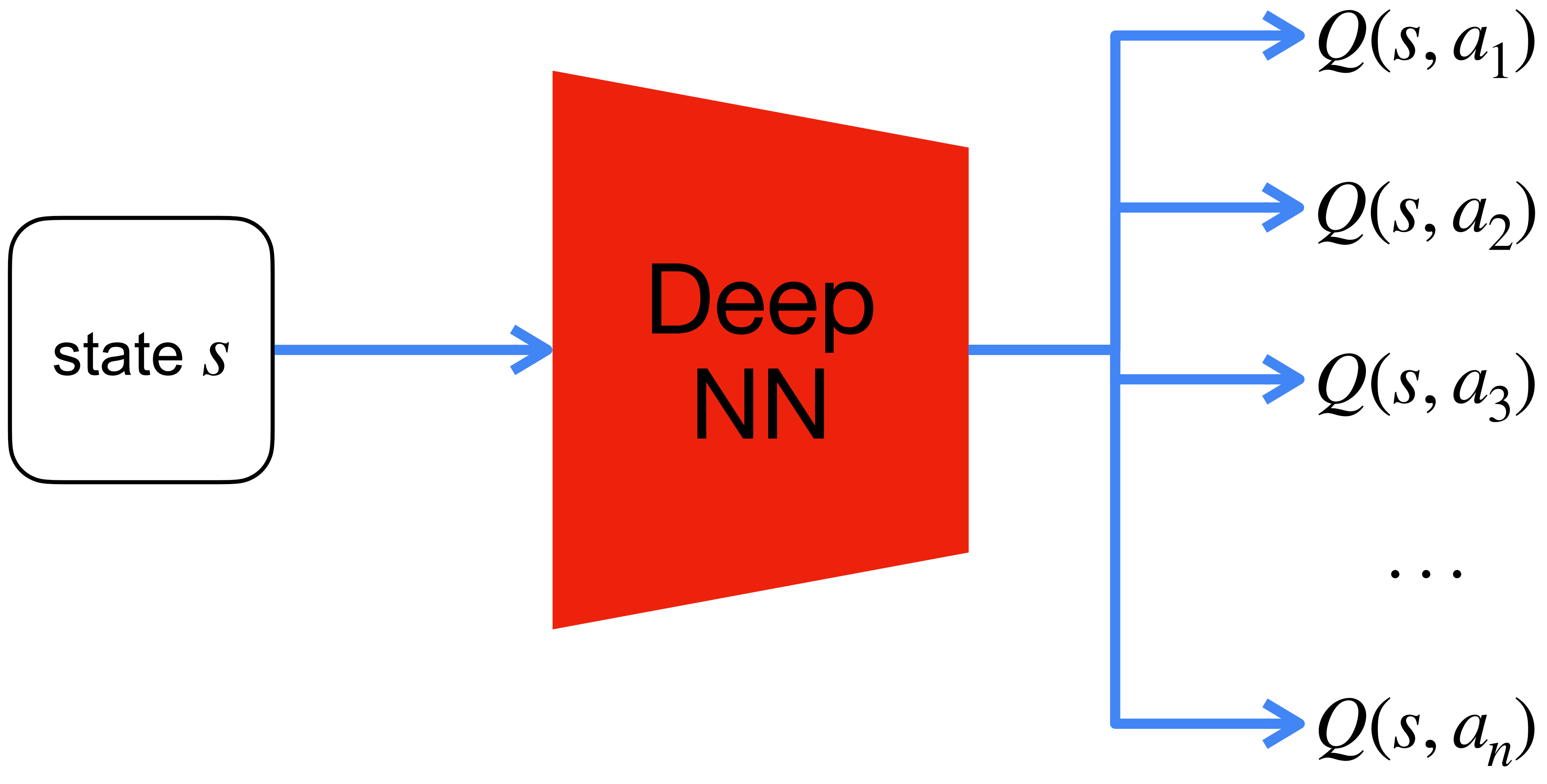
Main Idea: use neural networks to model Q-Function



Reinforcement Learning

Deep Q Network (DQN)

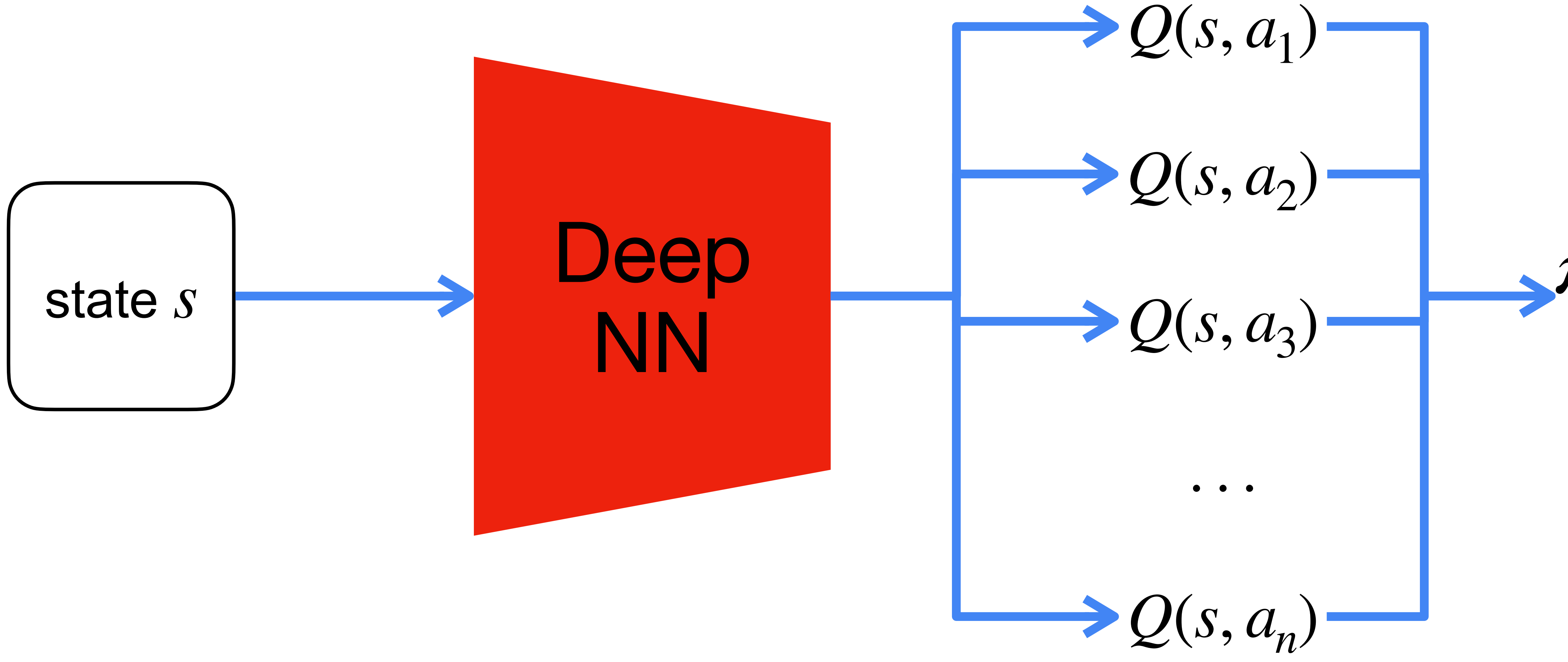
Main Idea: use neural networks to model Q-Function



Reinforcement Learning

Deep Q Network (DQN)

Main Idea: use neural networks to model Q-Function

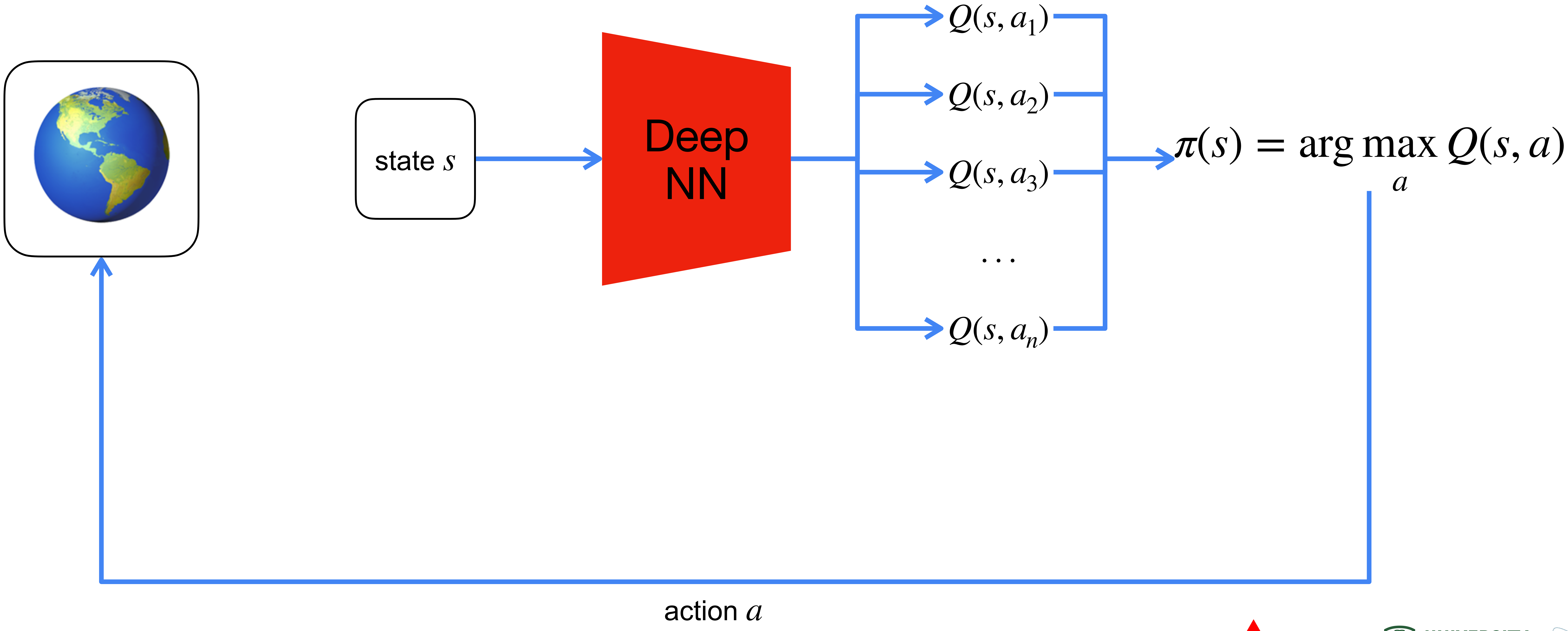


$$\pi(s) = \arg\max_a Q(s, a)$$

Reinforcement Learning

Deep Q Network (DQN)

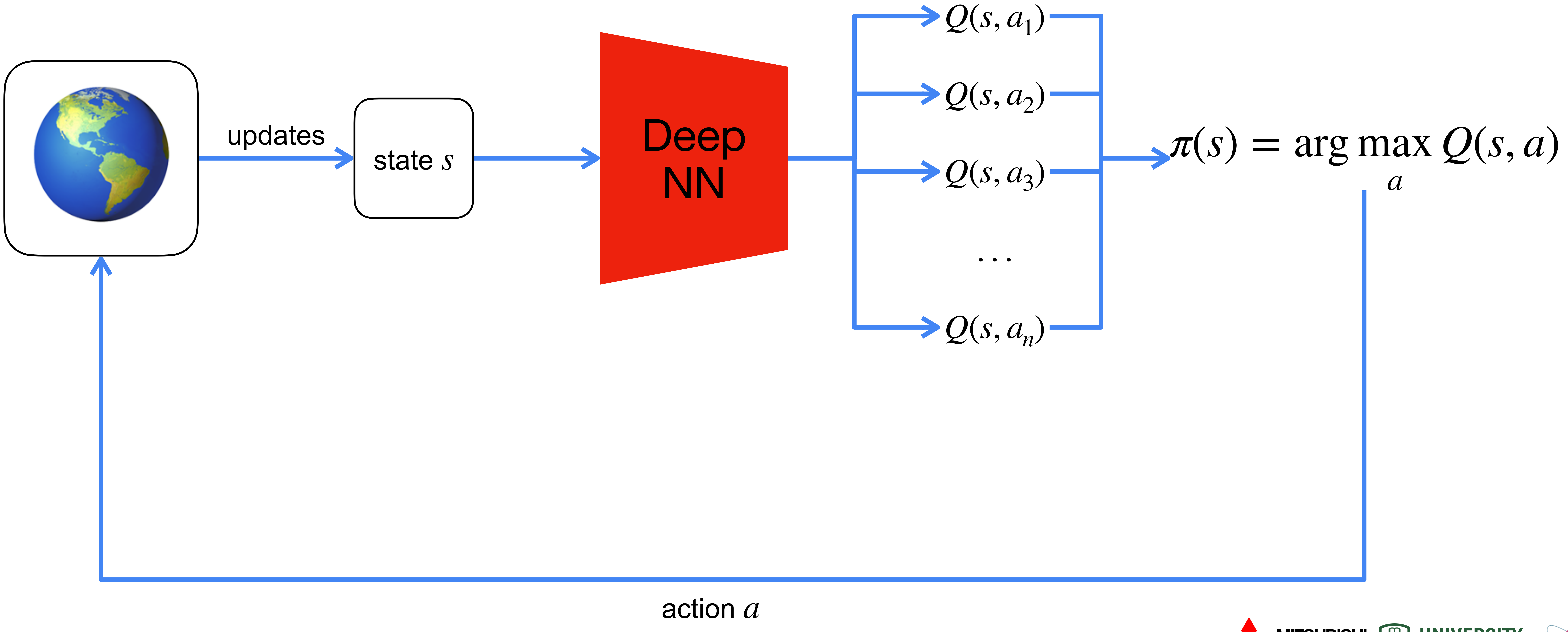
Main Idea: use neural networks to model Q-Function



Reinforcement Learning

Deep Q Network (DQN)

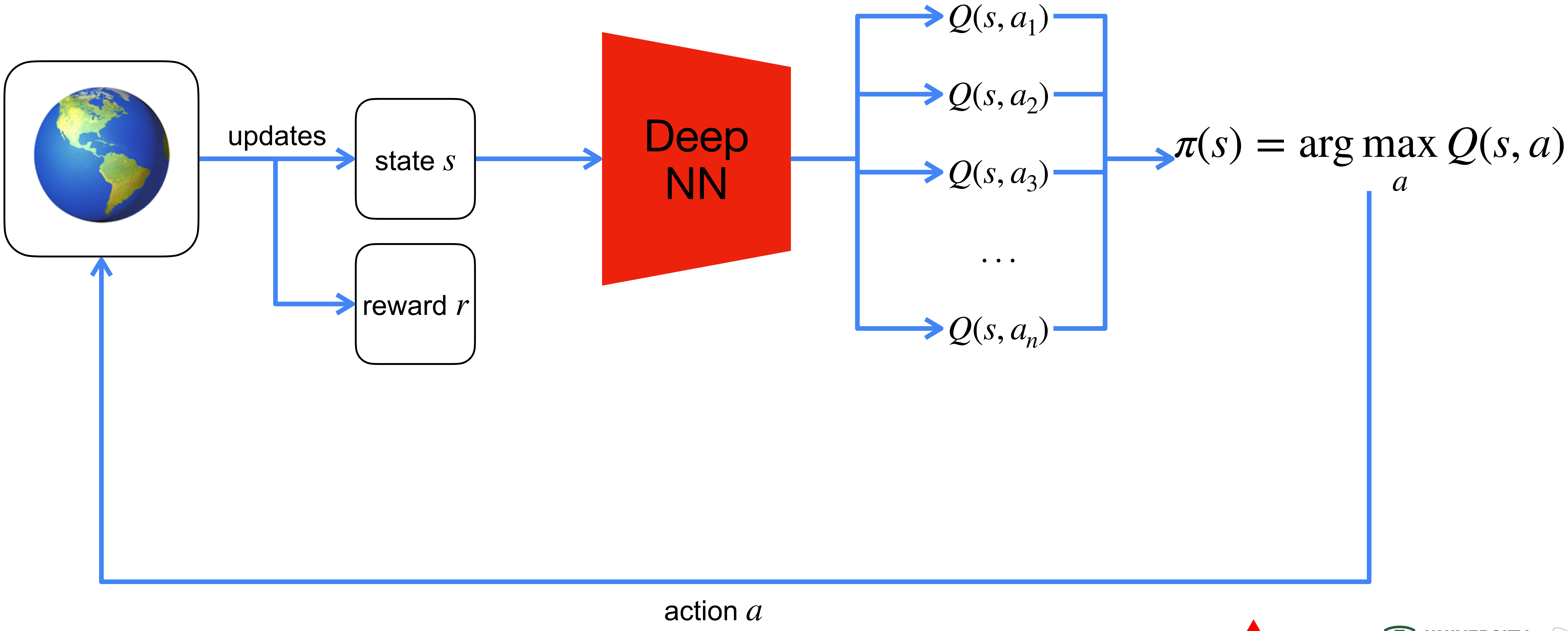
Main Idea: use neural networks to model Q-Function



Reinforcement Learning

Deep Q Network (DQN)

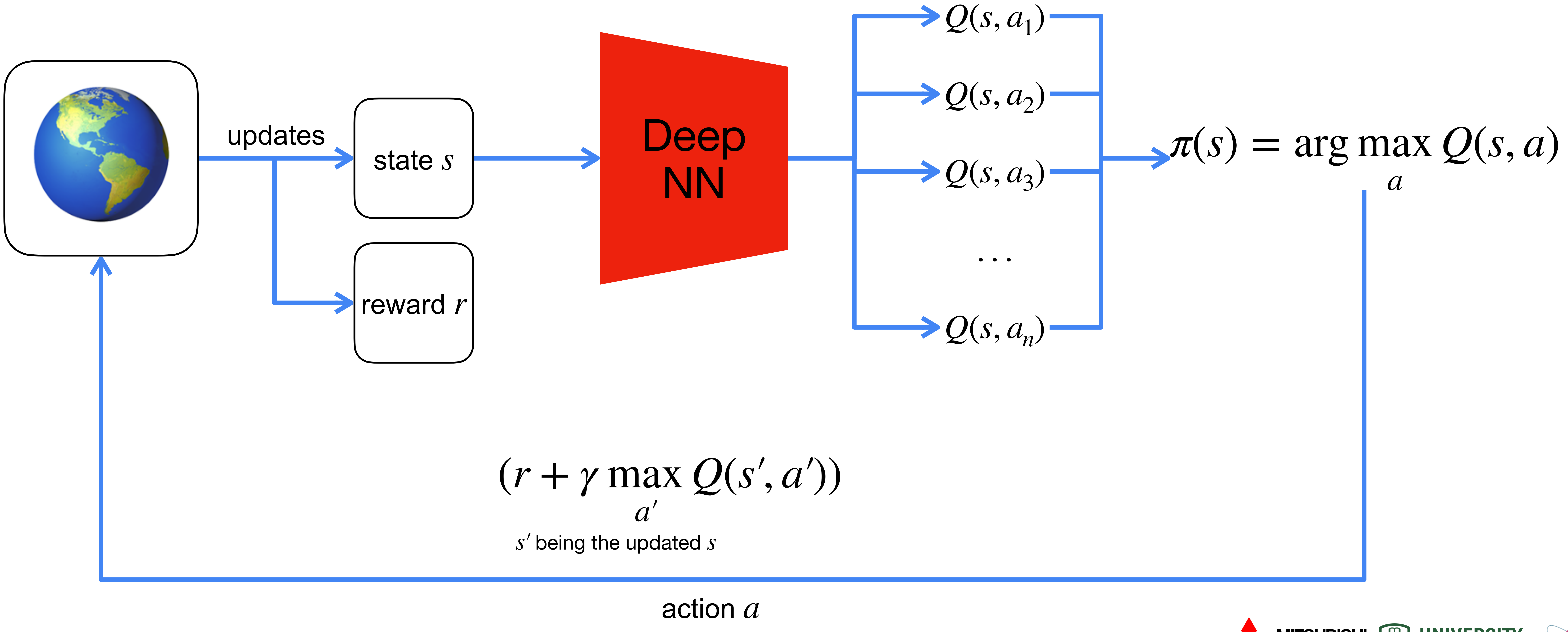
Main Idea: use neural networks to model Q-Function



Reinforcement Learning

Deep Q Network (DQN)

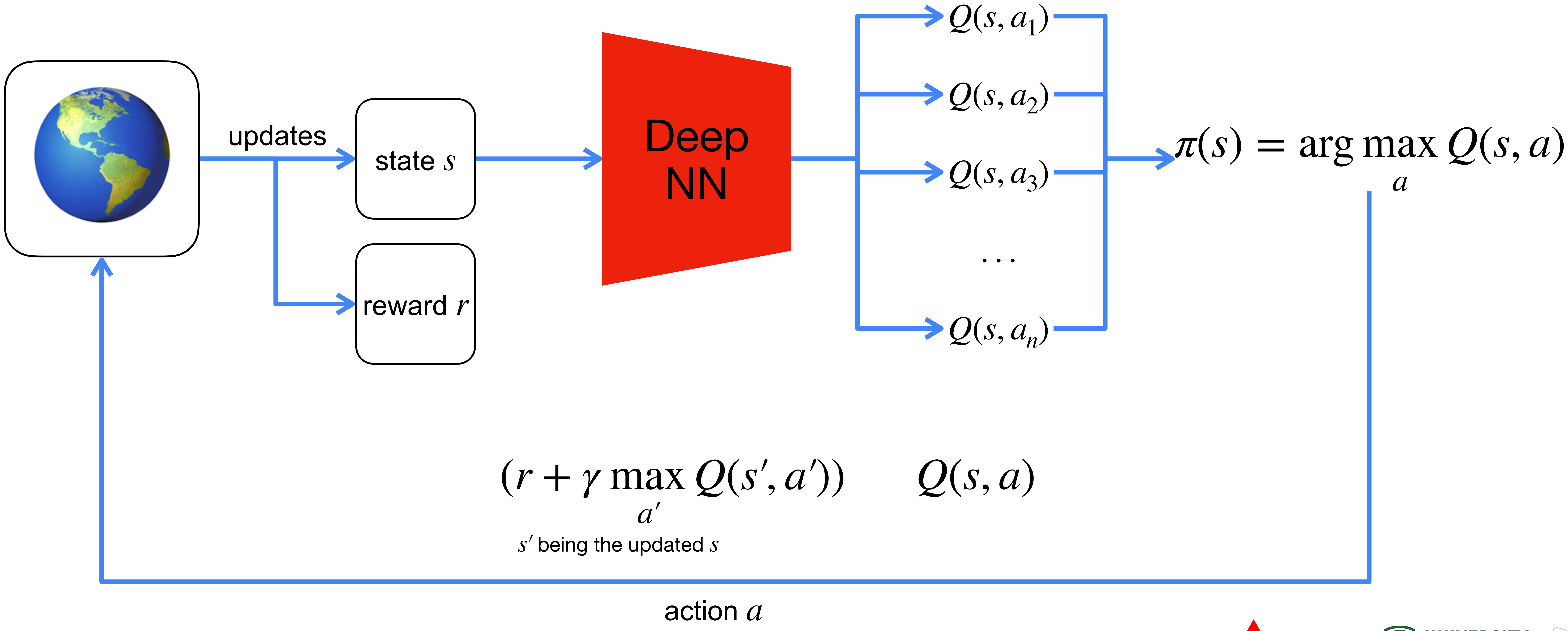
Main Idea: use neural networks to model Q-Function



Reinforcement Learning

Deep Q Network (DQN)

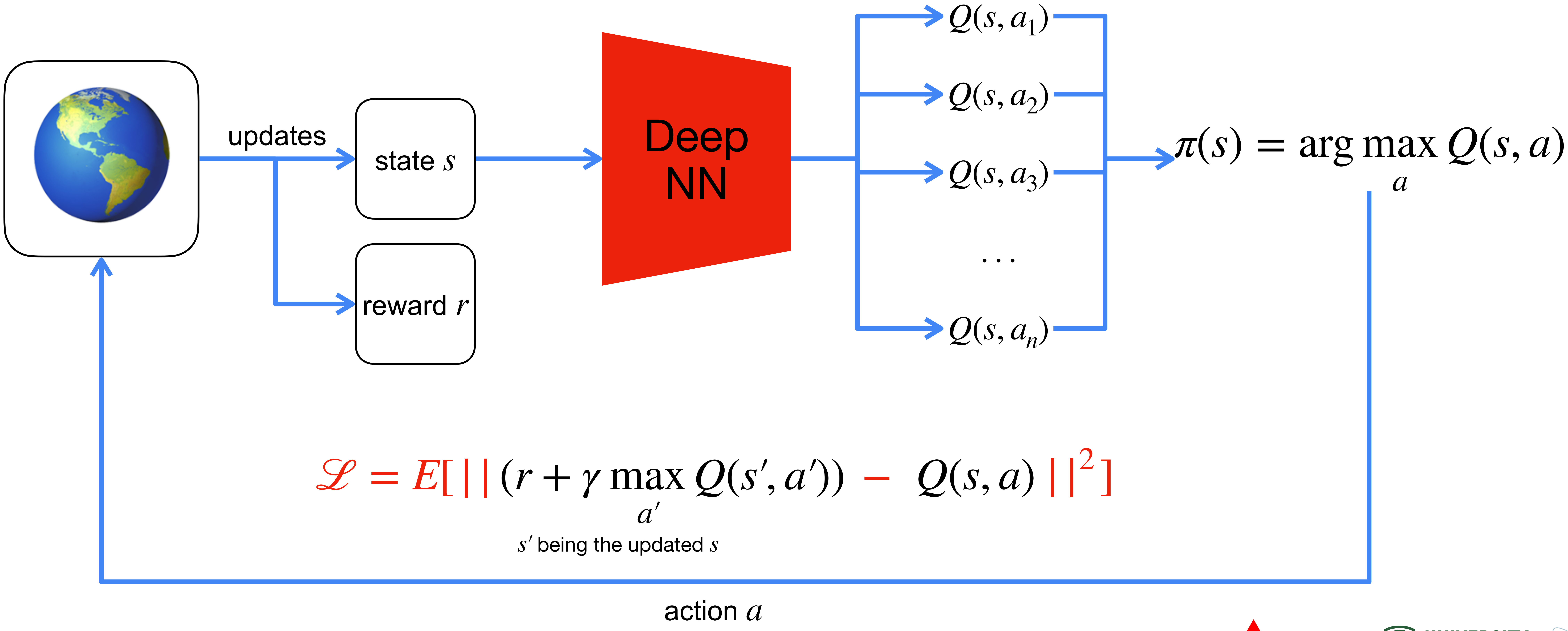
Main Idea: use neural networks to model Q-Function



Reinforcement Learning

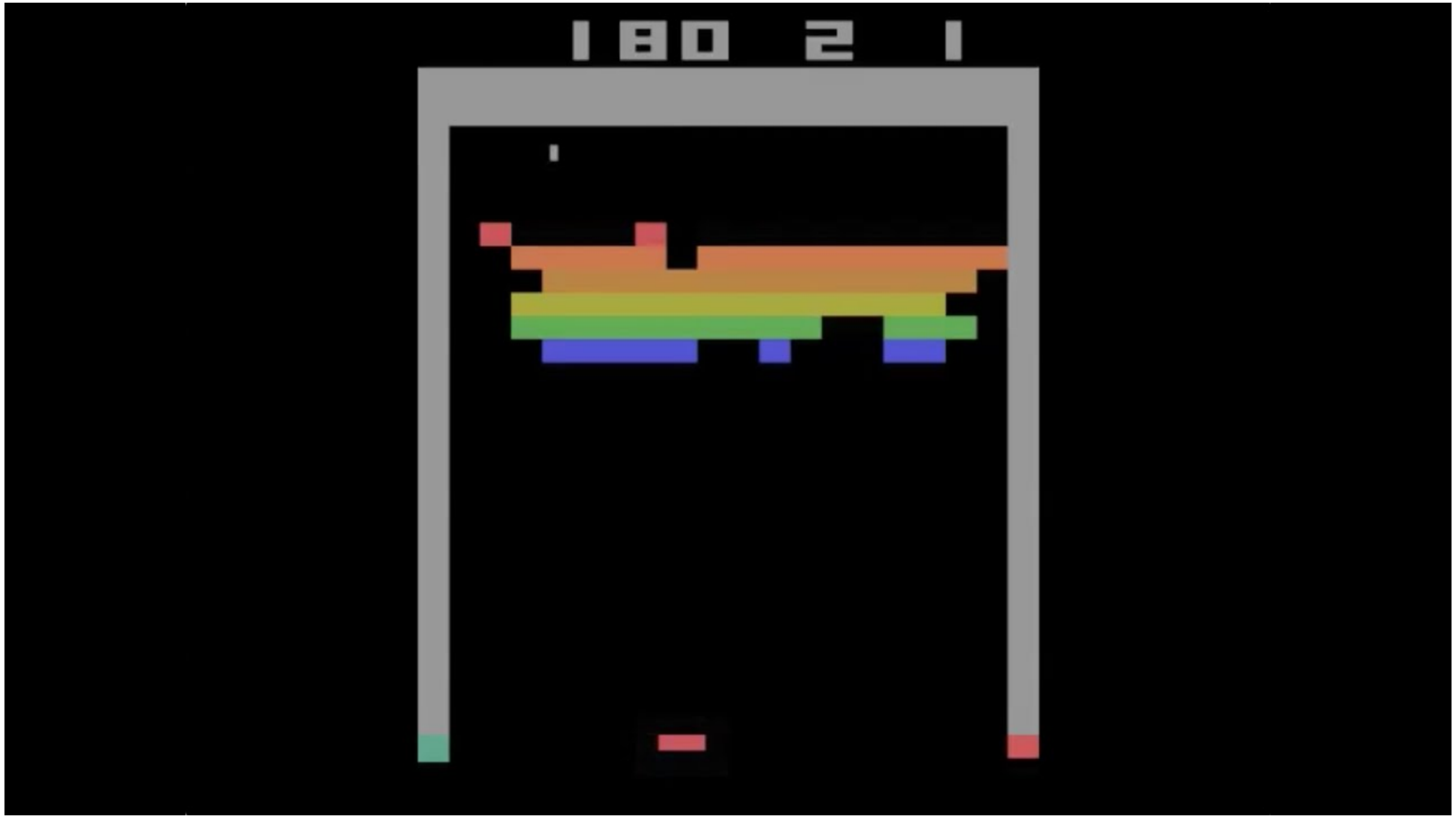
Deep Q Network (DQN)

Main Idea: use neural networks to model Q-Function



Reinforcement Learning

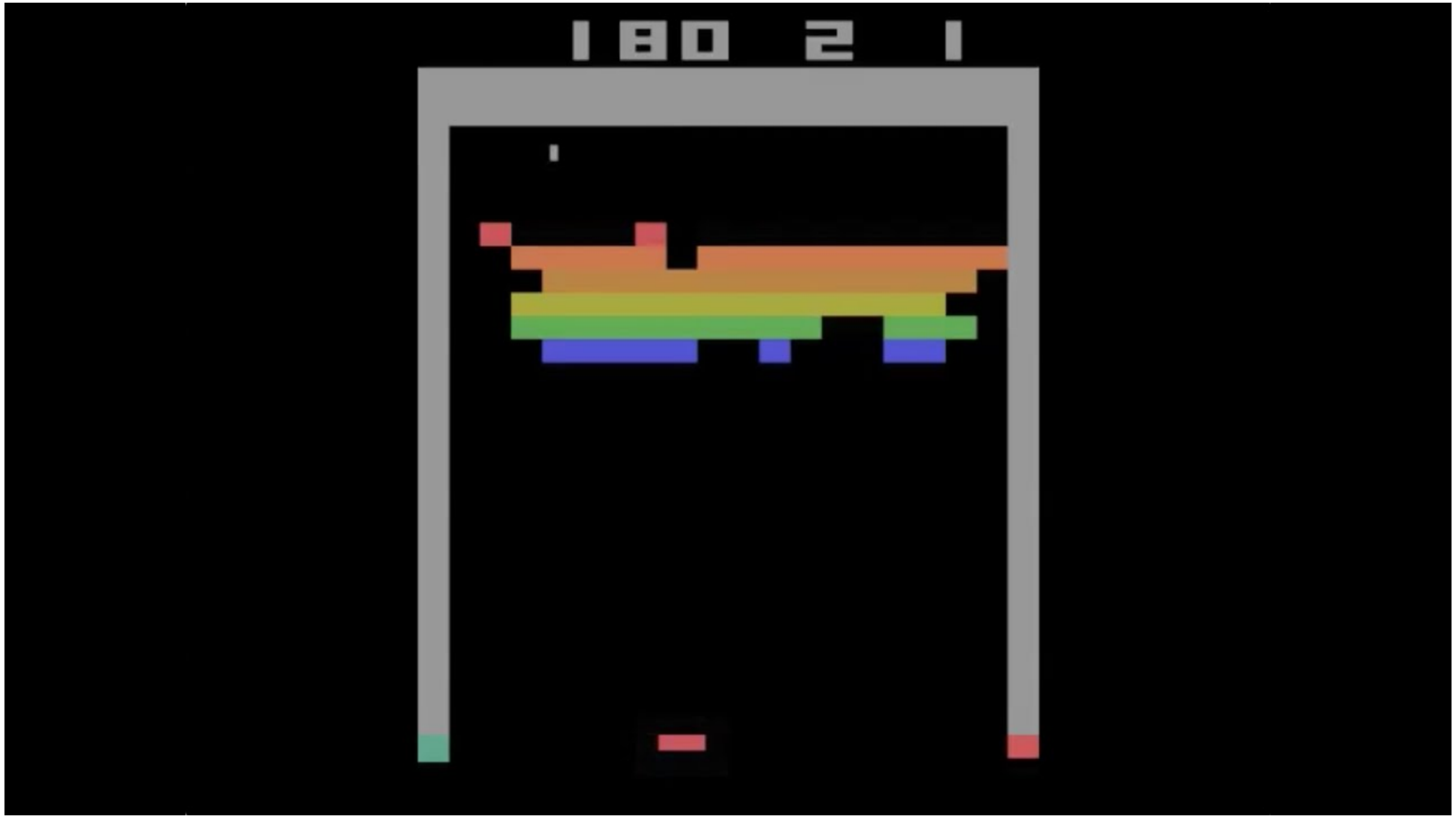
Real-World Applications



Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *nature* 518.7540 (2015): 529-533.

Reinforcement Learning

Real-World Applications



Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *nature* 518.7540 (2015): 529-533.

MITSUBISHI ELECTRIC UNIVERSITY OF ALBERTA amii

BO or RL?

Bayesian Optimization	Reinforcement Learning

BO or RL?

Bayesian Optimization

Reinforcement Learning

Environment

Static

Dynamic

BO or RL?

Bayesian Optimization		Reinforcement Learning
Static	**Environment**	Dynamic
Deterministic	**Action**	Stochastic

BO or RL?

	Bayesian Optimization	Reinforcement Learning
Environment	Static	Dynamic
Action	Deterministic	Stochastic
Reward	Direct and immediate	Delayed and/or Sparse

MITSUBISHI ELECTRIC UNIVERSITY OF ALBERTA amii