W05: Learning from Demonstration

Industrial tasks still performed by Humans

Manipulation tasks that require high dexterity

→ precise position and force control.

Tasks that are versatile with limited series.

Learning from Human Demonstrations: Principle

Transfer to the robot skills that took years for the humans to master.

Human can quickly re-train the robot to adapt to task changes.

The human teaches by showing how to perform the task.

Robotics and Autonomous Systems
Volume 57, Issue 5, 31 May 2009, Pages 469-483

A survey of robot learning from demonstration

Brenna D. Argall [a], Sonia Chernova [b], Manuela Veloso [b], Brett Browning [a]

Argall, Brenna | Faculty | Northwestern Engineering

Associate Professor in the School of Interactive Computing at Georgia Tech

Head of the Machine Learning Department at Carnegie Mellon University

Introduction

Policy: Mapping between states and actions

- A policy learning technique: Learning from Demonstration (LfD)

- Contrast to learning from experience e.g. Reinforcement Learning (RL) where data is acquired from exploration

- Related Fields: Neuroscience, psychology, linguistics, computer science

Support for LfD

- Traditional math-based approaches require perfect models, linearization and approximations.
- Reinforcement Learning (RL) requires domain specific expertise and it is hard to apply in real world.
- Learning from Demonstration (LfD) has a practical state-space. It does not require domain-specific expertise and it is intuitive

Problem Statement

- The world consists of states S and actions A, with the mapping between states by way of actions being defined by a probabilistic transition function $T(s' \mid s, a) : S \times A \times S \to [0, 1]$.
- We assume that the state is not fully observable.
- The learner instead has access to observed state Z, through the mapping $M : S \to Z$. A policy $\pi : Z \to A$ selects actions based on observations of the world state.
- We represent a demonstration $d_j \in D$ formally as k_j pairs of observations and actions: $d_j = \{(z_j^i, a_j^i)\}$, $z_j^i \in Z$, $a_{ij} \in A$, $i = 0 \cdots k_j$.

Problem Statement

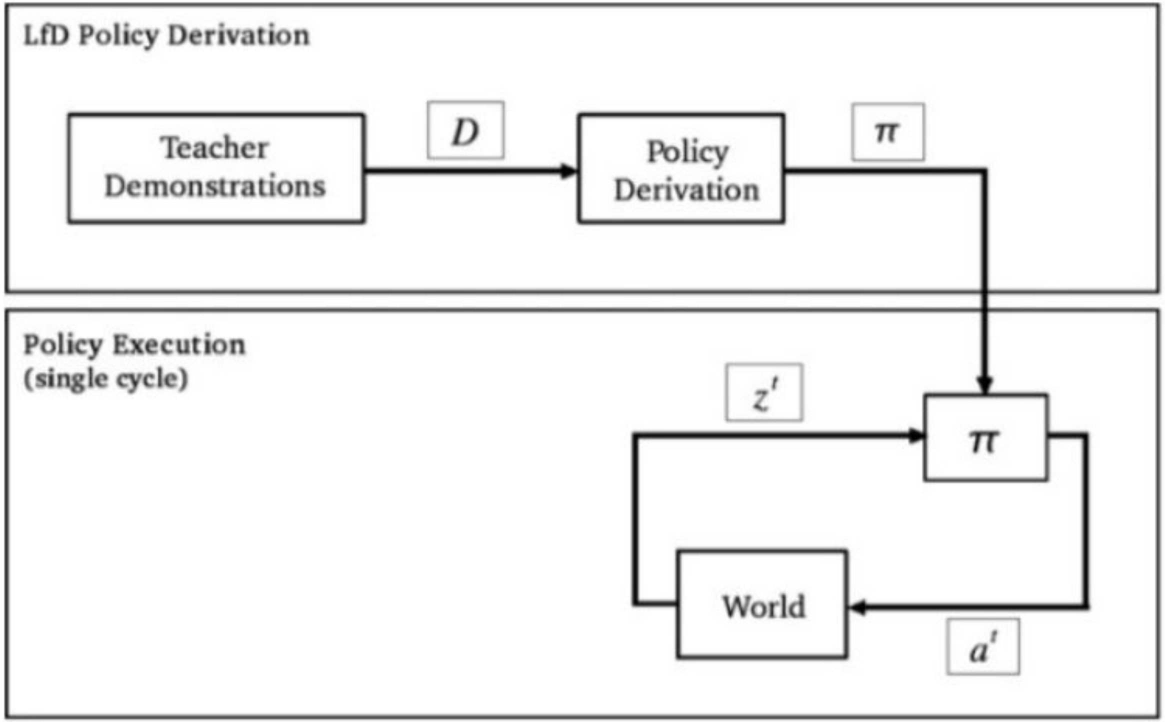


Fig. 1. Control policy derivation and execution.

Design Choices

Demonstration Approach

- Demonstrator
  - Human vs robot controller
  - Self vs external execution
- Demonstration Technique
  - Batch vs interactive
- Problem Space
  - Discrete vs continious state-space
  - Low-level/basic high-level/complex behavior actions

Gathering Examples

- How to record the data?
- Which platform to execute an action?

Correspondence

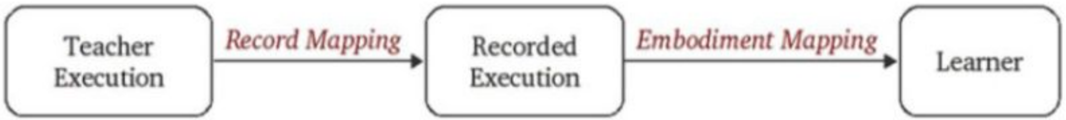
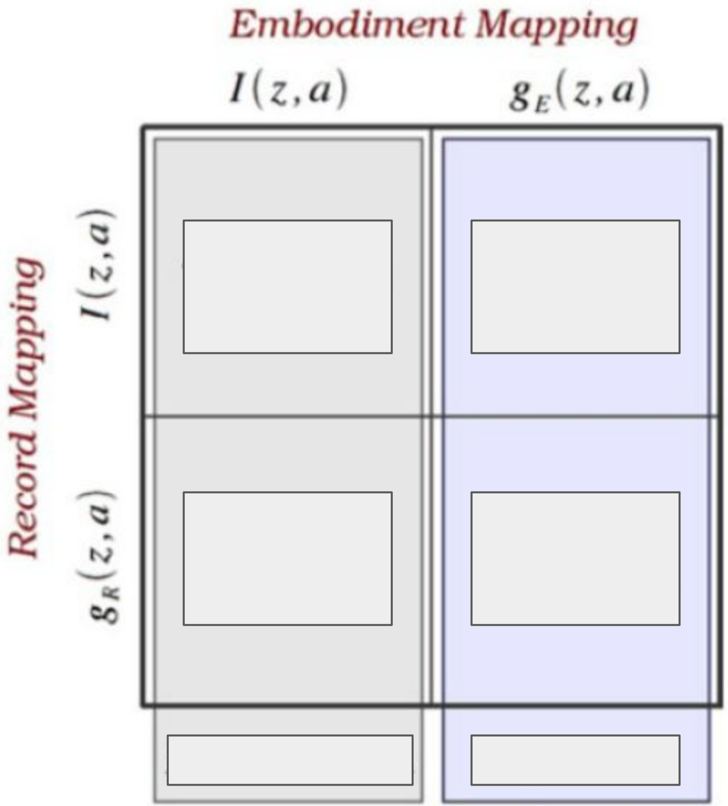


Fig. 3. Mapping a teacher execution to the learner.

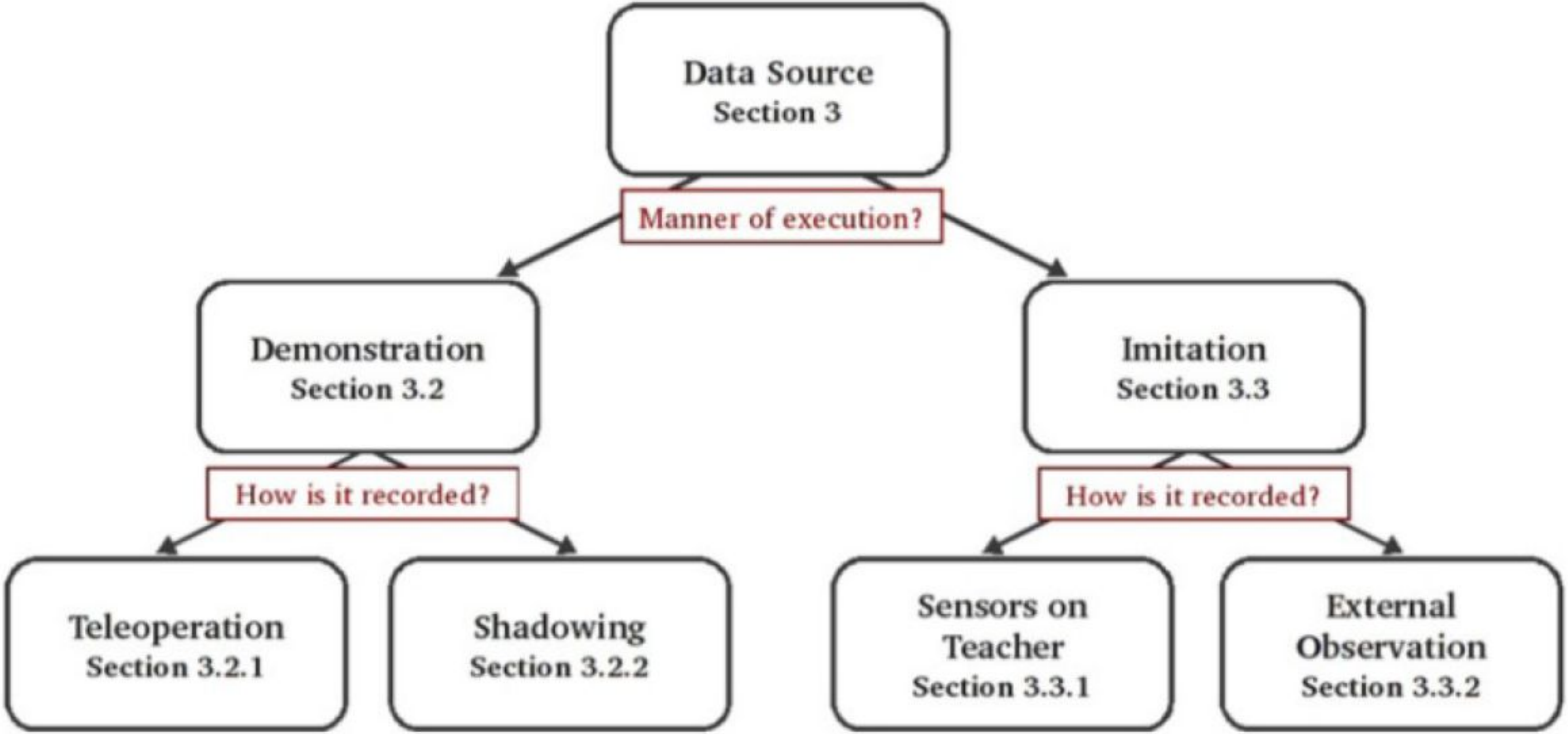


Basic Issues:

– Sensing

– Mechanics

Gathering Examples



Demonstration

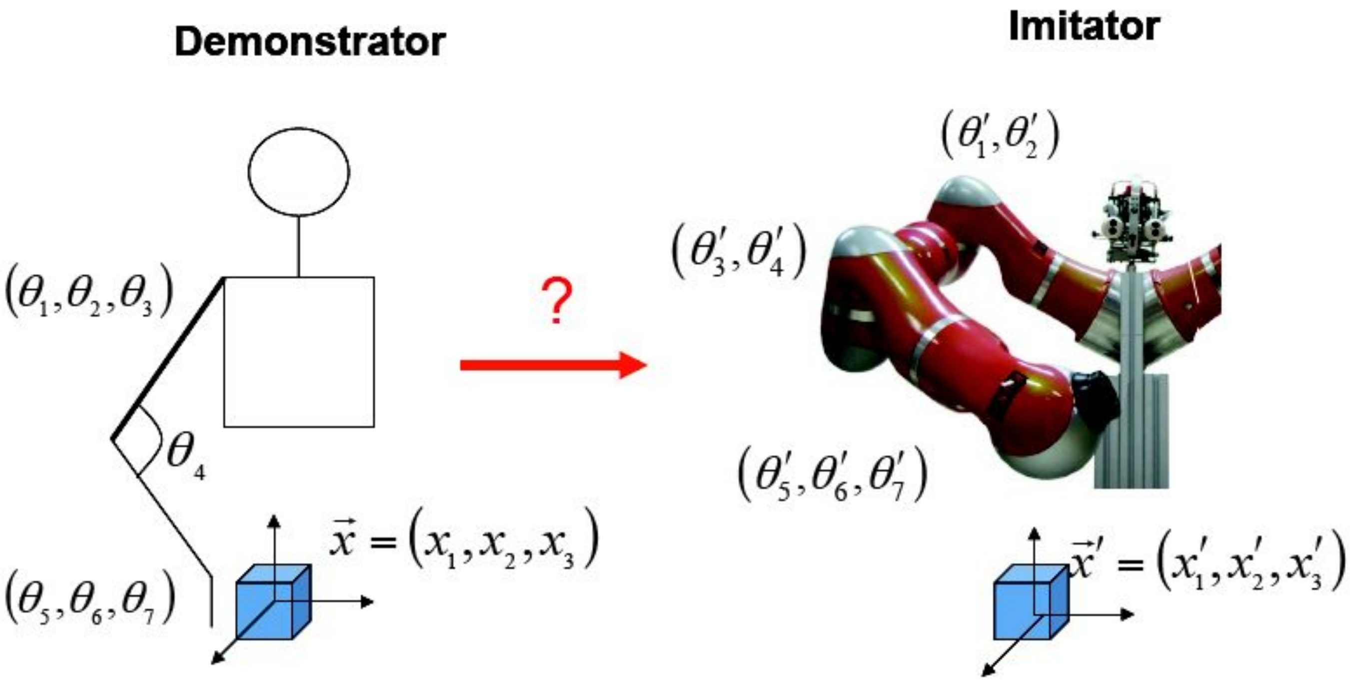
Teleoperation

- Direct record/direct embodiment
- Examples: helicopter controller, grasping,kinesthetic teaching, speech controller.

Shadowing

- Non-direct record/direct embodiment
- Record mimicking execution

Correpondence Problem



Establish a correspondence across degrees of freedom when feasible.

Which interface?

Kinesthetic Teaching:

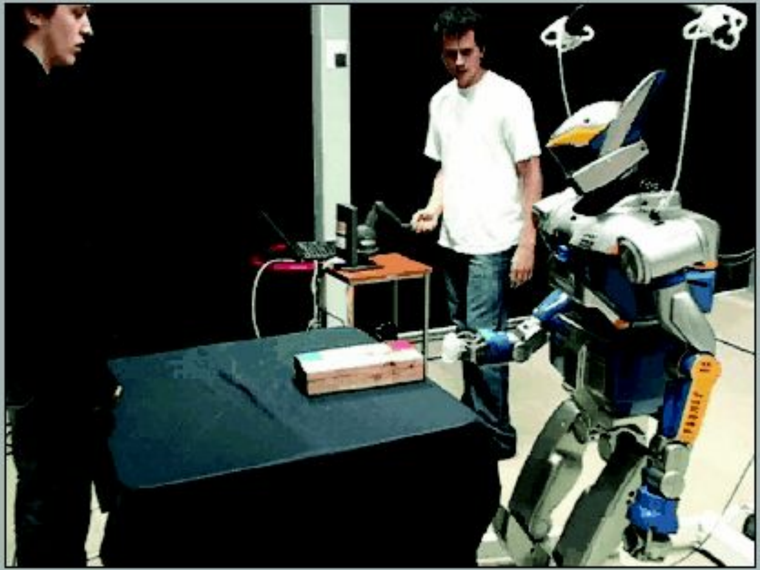
Pros:

- Solve correspondence problem
- Transmit kinematic & haptic information

Cons:

- Need two hands to teach movements of a few DOFs

Which interface?



Haptic devices:

Pros:
- Solve correspondence problem
- Transmit kinematic & haptic information

Cons:
- Requires training
- User far from task location

Imitation

Non-direct embodiment mapping

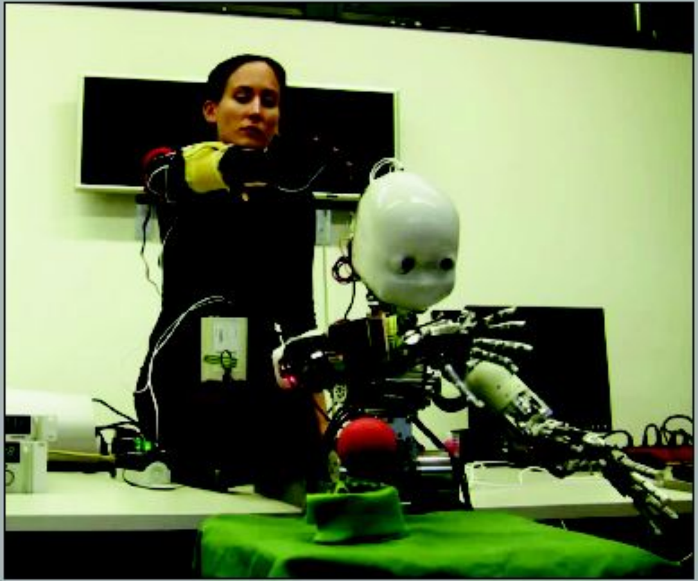
Sensors on teacher

- Limited applicability (wearable sensors etc.)

External observation

- Additional computational load to estimate action/state of the teacher

Which interface?



Motion sensors:

Pros:

- Real-time kinematic information
- Solve correspondence problem

Cons:

- Require to wear the system
- No haptic information

Which interface?

Vision:

Pros:
- Unobtrusive
- Record information on whole body.

Cons:
- Correspondence problem.
- No haptic information

Full body motion tracking using vision. Ude et al 2004

Other Approaches

- Record only states not actions
- Design low-level controllers for desired state transitions

Deriving a Policy

Three main approaches to derive a policy:

– Mapping Functions

– System Model

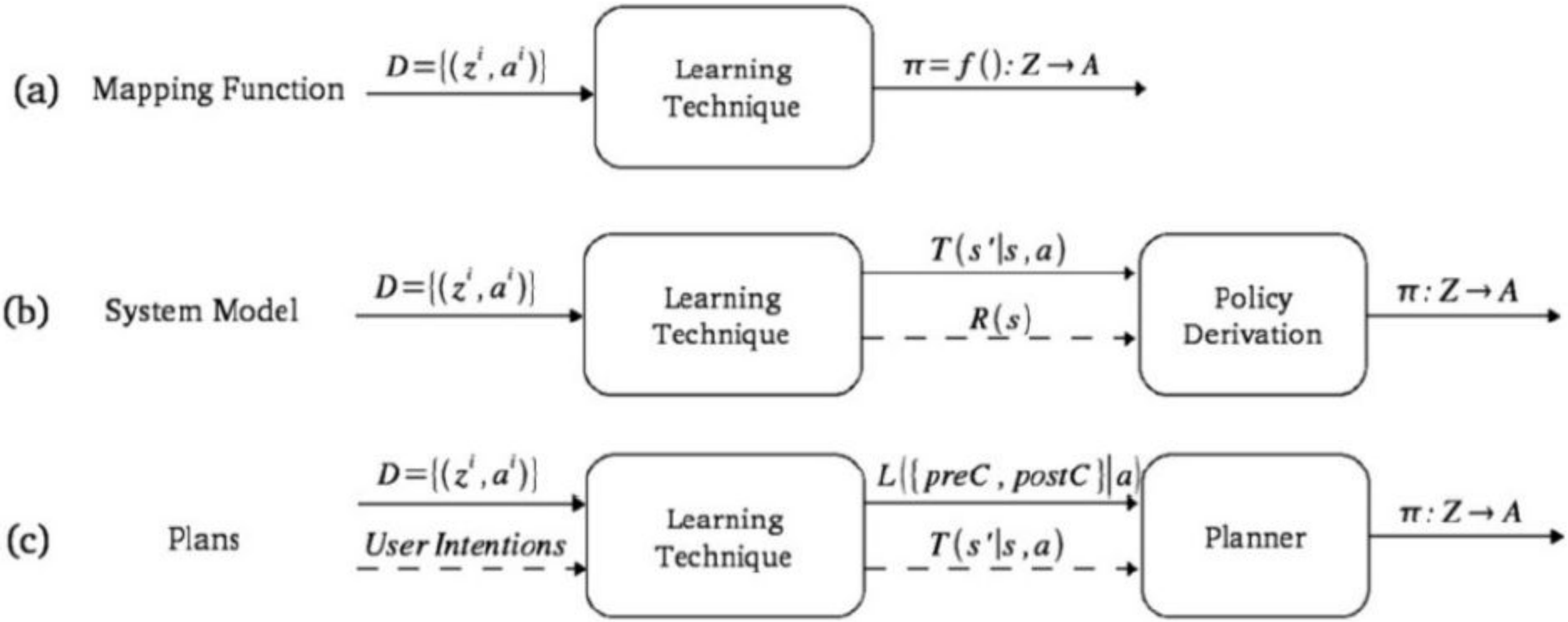
– Plans

Objectives:

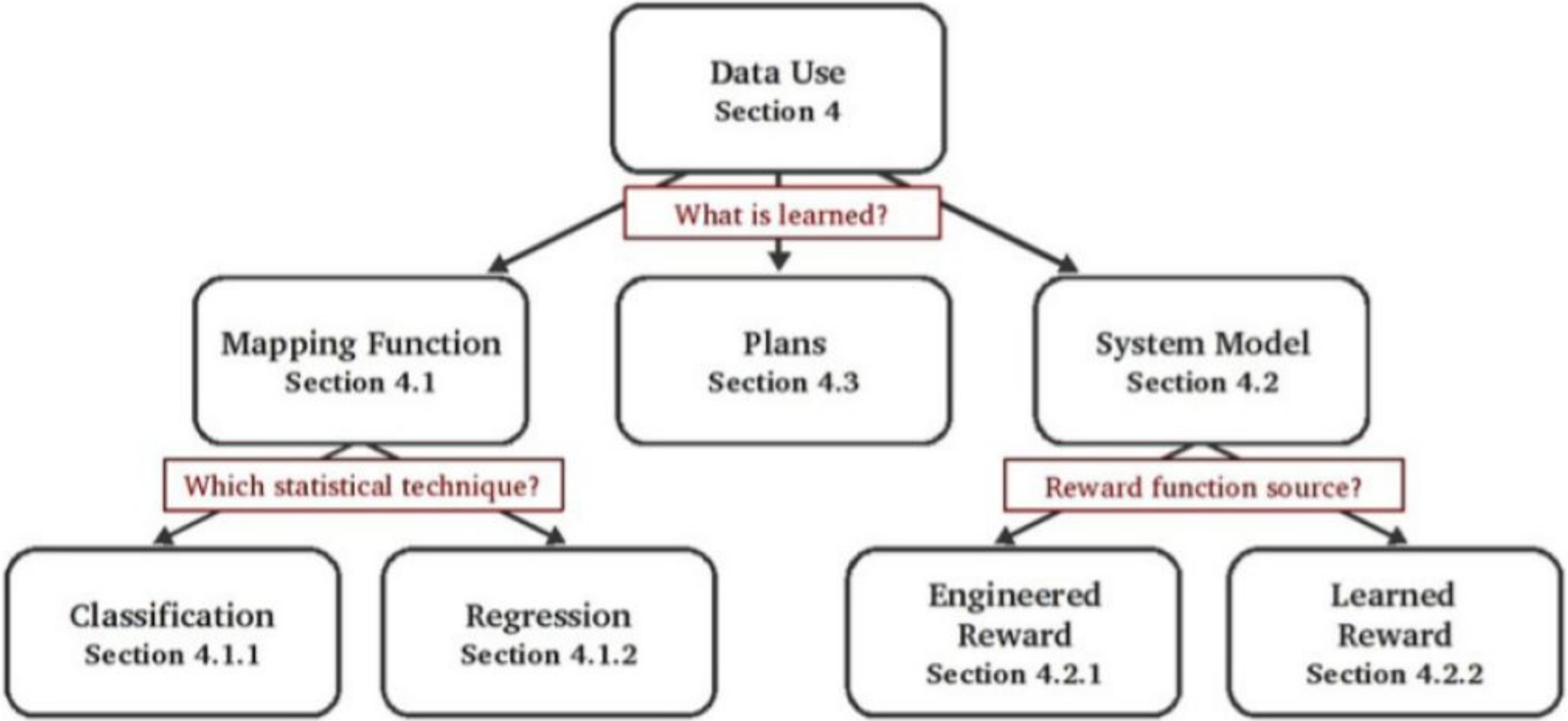
– Minimal parameter tuning

– Fast learning times with fewer iterations

Deriving a Policy



Deriving a policy



Mapping Functions

Approximates the state to action mapping, $f() : Z \rightarrow A$, for the demonstrated behavior

There are mainly two sub-approaches:

- Classification: Discrete output
- Regression
  - Continuous output
  - Typically applied for low-level actions

System Model

Uses a state transition model of the world, $T(s' \mid s, a)$ to derive a policy $\pi : Z \rightarrow A$.

● A reward function $R(s)$ which associates reward value r with world state s is either:

– Defined by the user or

– Learned from the demonstrations

Plans

Map states directly to actions is to represent the desired robot behavior as a plan.

● Pre-conditions: the state that must be established before the action can be performed

● Post-conditions: the state resulting from the action's execution

● Rely on annotations or intentions from the teacher