Lecture 2

CS 6380 Artificial Intelligence

Intelligent Agents

Department of Computer Science and Engineering

Indian Institute of Technology Madras (IITM), India

Some material based on lecture slides for CSE 571: Artificial Intelligence by Siddharth Srivastava, ASU.

Logistics and Recap

Resources

- Course Website: https://hails-group.github.io/courses/cs6380/jn26.html
  - Slides posted here
  - Schedule already available

- Announcements: Moodle

- Course Email: pulkitv@cse.iitm.ac.in
  - Subject: [CS6380] <Subject>
  - No URGENT email responses. Please **plan** ahead.
  - We will respond to each email within 48 hours.
  - Any course-related query should be posted on Moodle so that others can also benefit from the answer.

Exams

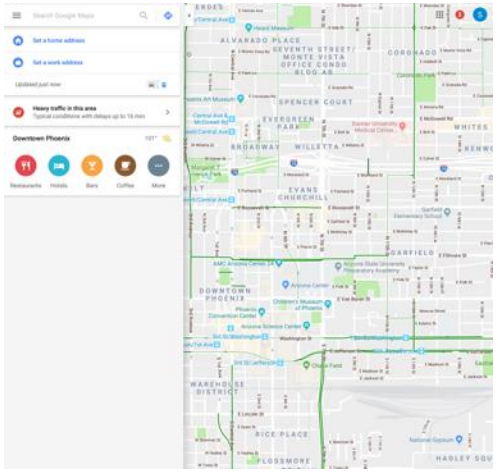
- 3 Exams:
  1. [Quiz 1] Aug 31
  2. [Quiz 2] Oct 12
  3. [End Sem] Nov 11

- More details closer to exams.

Grading Breakup

- Exams: 70%
  1. Quiz 1 : 20%
  2. Quiz 2 : 20%
  3. End Sem: 30%
- Assignment-Quizzes: 25%
- In-class Quizzes: 5%
- Relative Grading

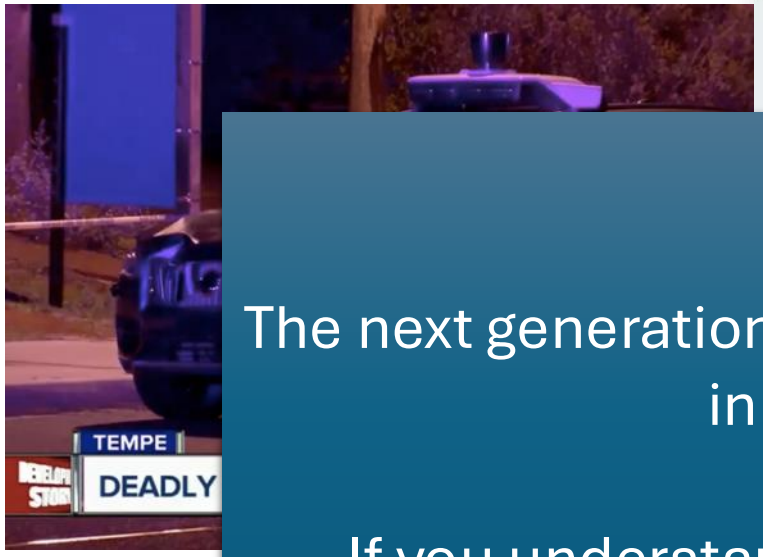
Numerous applications, success stories for AI

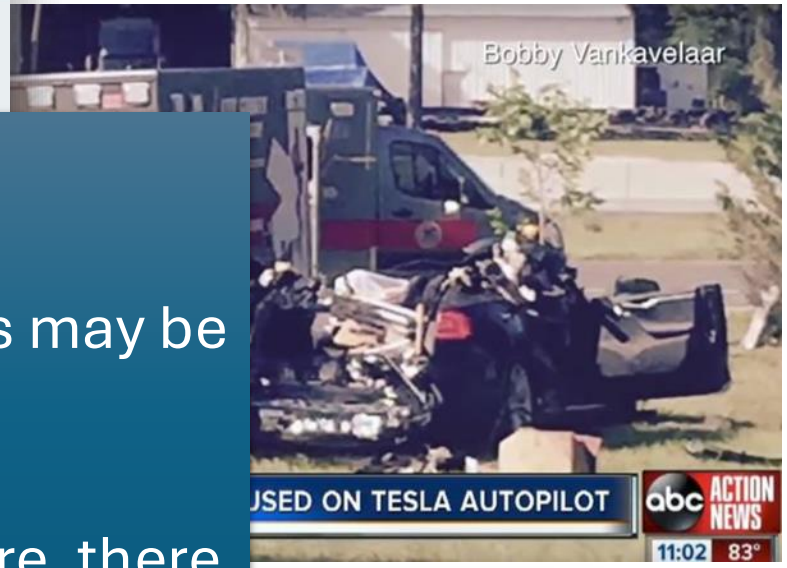


Contemporary Robots

https://www.roboticsbusinessreview.com

A More Sober Look at Current State of AI



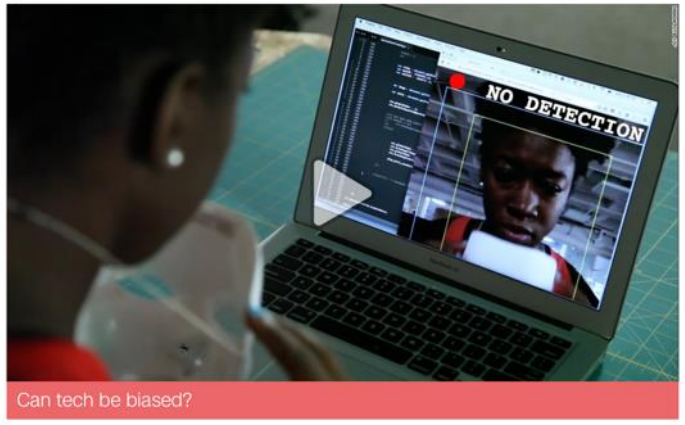


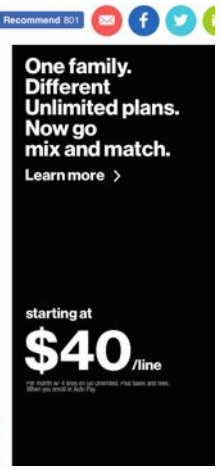
Good News!!

The next generation of AI designers and engineers may be in this classroom today

If you understand the concepts presented here, there will be fewer bad designs!

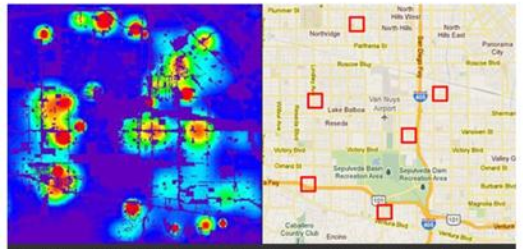
AI is hurting
Experts want
by Heather Kelly @heatherkelly
July 23, 2018: 10:14 AM ET





elligence Is Now
Used to Predict Crime. But Is
It Biased?

The software is supposed to make policing more fair and accountable. But critics say it still has a way to go.

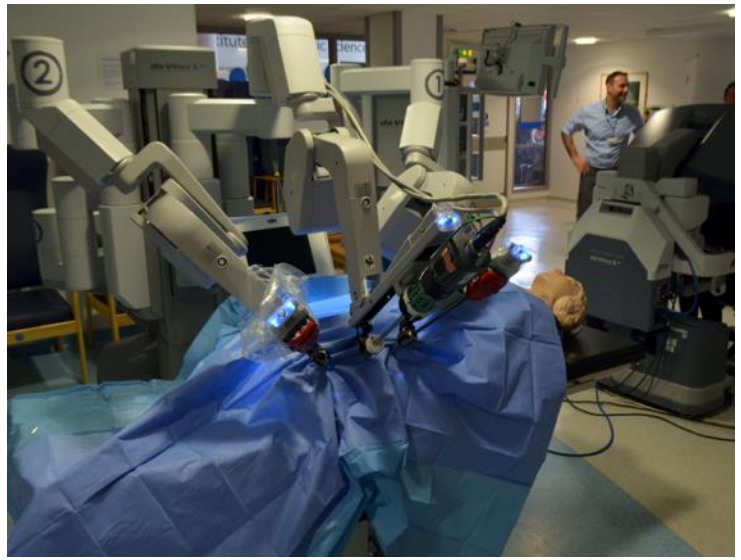


By Randy Rieland

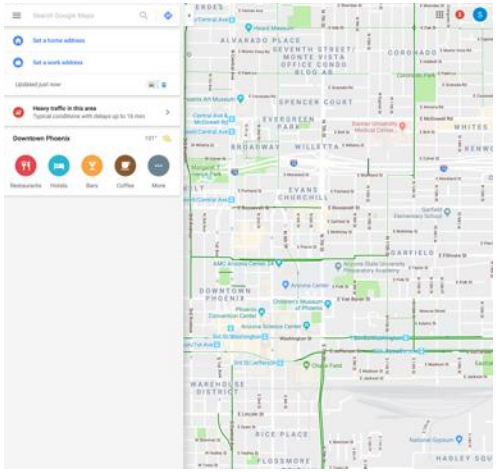
Agents and Their Behavior

“Agents”

https://www.roboticsbusinessreview.com



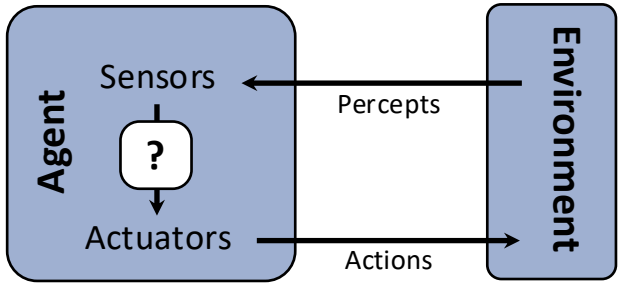
More examples…



Can we formalize the AI problem in a way that captures both forms (physical and online) of agents?

What is an AI Agent?

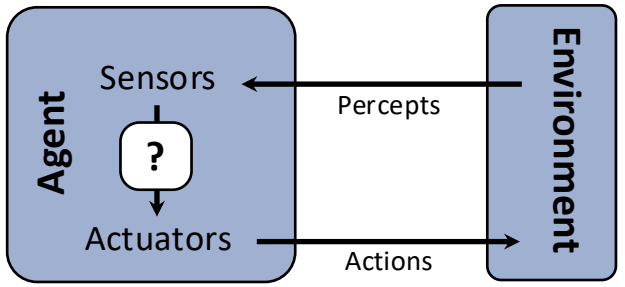
- An **agent** is the entity that AI algorithms control
  - it *perceives* and *acts*.
- Different assumptions about sensors, actuators, environments define different types of agents
  - Netflix, paper matching system, iPad face scanner, roomba household robot, …
- Different types of agents amount to different types of problems
- **This course** is about:
  - General AI techniques for a variety of problem types
  - Learning to recognize when and how a new problem can be solved with an existing technique



"States"

Abstract Notion of an Agent

- Agent function
  - Maps percept histories to actions:
  - $f: P^* \rightarrow A$

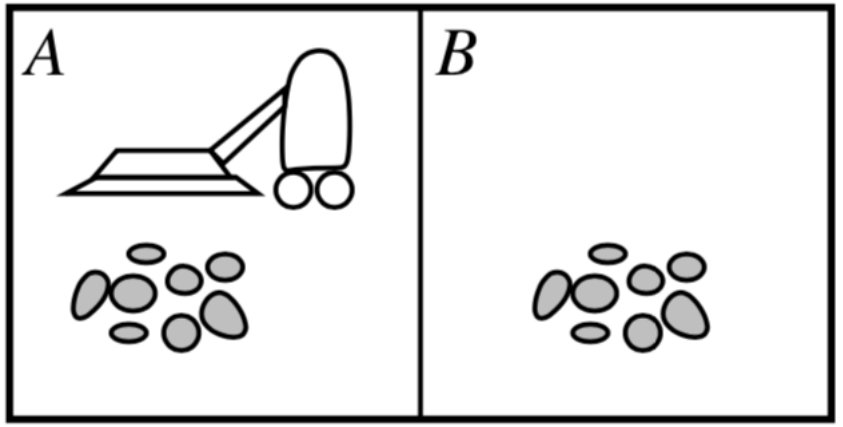


- Agent function needs to be computed
  - Agent program: the program that computes f for a given agent and agent function

- Agent behavior is encoded by the agent function

Vacuum-cleaner World

- Percepts
  - Information returned by the agent's sensors
  - VC world: location, contents, E.g. <A, Dirty>
- Actions
  - What the agent can do
  - Left, Right, Suck, NoOp
- Example of agent function:



function REFLEX-VACUUM-AGENT([*location*,*status*]) **returns** an action

if *status* = *Dirty* **then return** *Suck*
else if *location* = *A* **then return** *Right*
else if *location* = *B* **then return** *Left*

Agent Function vs. Agent Program

- The **agent function** is a mathematical abstraction
  - Maps every possible percept sequence to an action: $f : P^* \rightarrow A$
  - An external description of behavior
  - Usually infinite; we can describe it, but we cannot store it

- The **agent program** is the concrete implementation
  - Runs inside the agent, on physical hardware
  - Takes only the **current percept** as input, not the whole sequence
  - If history matters, the program must maintain internal state itself

- **agent = architecture + program**
  - Architecture: the sensors, actuators and computing device the program runs on
  - The program must be appropriate for the architecture: a program that recommends Walk needs legs

Why Not Just Tabulate the Agent Function?

- A **table-driven agent** stores the agent function explicitly
  - Append the new percept to the percept sequence, look it up, return the action
  - It is a correct implementation of any agent function, and it is hopeless

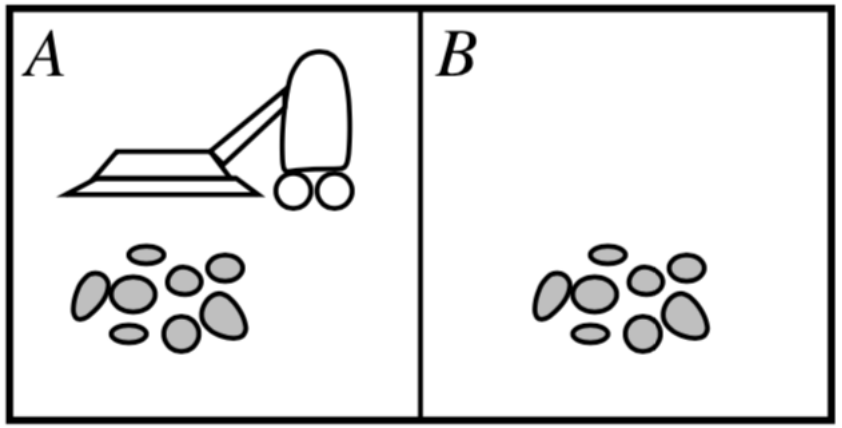
- The table has $\Sigma_t |P|^t$ entries for a lifetime of T steps
  - Chess, with a tiny visual input: more than 10^{150} entries
  - A taxi camera at 27 MB/s: on the order of $10^{250,000,000,000}$ entries per hour of driving
  - There are roughly 10^{80} atoms in the observable universe

- So the table fails on every count
  - No physical agent has the storage
  - No designer could ever write the entries
  - No learner could ever fill them in from experience

- **The central question of AI: how do we produce rational behavior from a small program?**

Reflex Agents

- Follow recipes or pre-coded behavior
  - Condition-action rules *mapping current percepts* to action
  - Minimal computation
- Different courses of action are not considered
- Many expert systems are designed in this way



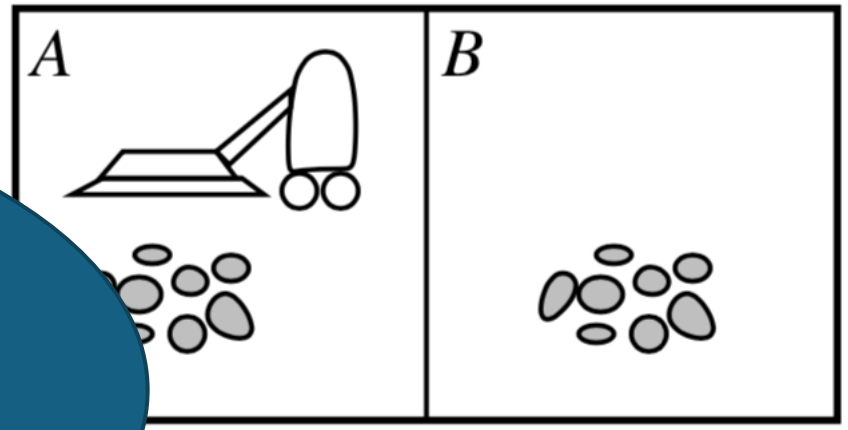
function REFLEX-VACUUM-AGENT([*location*,*status*]) **returns** an action

if *status* = *Dirty* **then return** *Suck*
else if *location* = *A* **then return** *Right*
else if *location* = *B* **then return** *Left*

Partial Observability: Reflex Agents with State

- Percepts: contents of current location
  - Sensors do not report cu
- What would a refl
  - If dirty, Suck
  - If clean, Left? R
- What could a reflex a
  - If start: Left; <record tried_lef
  - If *not start and tried_left*: <record location=A>
  - If not start and tried_left: <record location=B>
  - If not start and location=A and clean: moveRight

  …



Can we compute such functions?
Need a way to compare possible behaviors

Performance Measure

- Executing an agent function generates a sequence of environment states
- Given a utility function/goal condition, this state sequence can be evaluated
  - Expected values in case of stochastic environments

Designing Performance Measures

- Measure **states of the environment**, not the agent's behavior
  - Otherwise the agent can score well in ways you never intended

- The classic failure: reward the vacuum agent for **amount of dirt sucked**
  - Optimal policy: suck up the dirt, dump it back on the floor, suck it again
  - Repeat forever. The score is unbounded; the floor is never clean
  - Better: reward a **clean floor**, per square, per time step

- Specify what you actually want, not how you think the agent should behave
  - Penalise electricity, noise and time if those matter to you
  - Whatever you leave out of the measure, the agent is free to ignore

- Even "clean floor" hides a choice
  - Average cleanliness over time? Or the worst moment? Or minimum over squares?
  - Reckless-but-usually-fine vs. slow-but-never-bad are different measures

Rationality: A Definition

- **For each possible percept sequence, a rational agent selects an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.**

- Rationality at any given moment depends on exactly four things:
  - **1. The performance measure** that defines the criterion of success
  - **2. The agent's prior knowledge** of the environment
  - **3. The actions** the agent can perform
  - **4. The agent's percept sequence** to date

- Change any one of the four and the rational action can change
  - Rationality is not a property of the program alone
  - It is a property of the **(agent, environment, performance measure)** triple

Rationality Is Not Omniscience, Perfection, or Clairvoyance

- An **omniscient** agent knows the actual outcome of its actions in advance
  - Impossible in any real environment

- You are crossing an empty street. Halfway across, a cargo door falls off a passing aircraft and flattens you.
  - Was crossing the street irrational?
  - No. Nothing in your percepts suggested it. The outcome was bad; the decision was not

- Rationality maximises **expected** performance; perfection maximises **actual** performance
  - We can demand the former from a design. The latter is not achievable
  - Rationality does not require predicting the unpredictable

- But this is not an excuse
  - If you cross without looking, and get hit, that **was** irrational
  - Failing to gather cheap, available information is a failure of rationality

Is the Simple Vacuum Agent Rational?

- **Fix the task environment:**
  - Performance: +1 for each clean square at each of 1000 time steps
  - The geography is known a priori; dirt distribution and start location are not
  - Suck cleans the current square; Left / Right move, and are no-ops at a wall
  - The agent perceives its location and whether that square is dirty

- Under **these** assumptions, the simple reflex agent is rational; no agent can do better

- Now change one assumption at a time:
  - **Penalise each movement** → once both squares are clean the agent should stop; ours oscillates forever and is no longer rational
  - **Let dirt reappear** → the agent should keep patrolling; stopping is now irrational
  - **Unknown geography, more squares** → the agent must explore and remember; a memoryless reflex program cannot be rational

What Rationality Demands: Gather, Explore, Learn, Be Autonomous

- **Information gathering:** acting to change what you will perceive next
  - Looking both ways before crossing; a doctor ordering a test
  - Doing this is required by rationality, because it maximises expected value

- **Exploration:** required when the environment is initially unknown
  - The vacuum agent dropped into an unmapped building must map it

- **Learning:** the agent must modify itself from what it perceives
  - Initial configuration encodes prior knowledge; experience should revise it
  - A dung beetle plugs its nest with a pellet. Remove the pellet and it keeps miming the action. It cannot learn, so it cannot be rational off-script

- **Autonomy:** relying on your own percepts, not only the designer's assumptions
  - An agent that ignores its percepts lacks autonomy and will fail when the designer's assumptions break
  - In practice: ship some prior knowledge, plus the ability to learn

Most of AI: Computing Agent Functions

- A perfectly rational agent uses the best possible agent function
  - Sometimes, computing the best agent function is not feasible!
  - UUV/UAV with limited battery power
  - Try to compute approximately optimal, or bounded rational agent functions
    - Optimal for that agent architecture/resources

- To develop algorithms for computing agent functions, we need to specify several aspects of the problem:
  - Performance measure
  - Environment
  - Actuators
  - Sensors

PEAS: Specifying a Task Environment

- Before designing any agent, specify the **task environment** as fully as possible

- **P: Performance measure**
  - What counts as success, stated in terms of environment states
- **E: Environment**
  - Everything relevant to the agent's decision-making
- **A: Actuators**
  - What the agent can do to the environment
- **S: Sensors**
  - What the agent can perceive of the environment

- PEAS is the design contract for the whole problem
  - Get it wrong and no amount of algorithmic cleverness will rescue the agent
  - Every algorithm in the rest of this course assumes a PEAS specification has already been fixed

PEAS Example: Automated Taxi Driver

- **Performance measure**
  - Safe, fast, legal, comfortable trip; maximize profits; minimize impact on other road users
- **Environment**
  - Roads, other traffic, police, pedestrians, customers, weather
- **Actuators**
  - Steering, accelerator, brake, signal, horn, display or speech to the passenger
- **Sensors**
  - Cameras, radar, speedometer, GPS, engine and fuel sensors, accelerometer, microphone, touchscreen

- Two things worth noticing:
  - The performance measure is **open-ended**, and its components **conflict**. Fast trades against safe; profit trades against comfort
  - Resolving those conflicts is a design decision, made before any code is written

PEAS Across Different Agent Types

Agent Type	Performance	Environment	Actuators	Sensors
Medical diagnosis	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses	Keyboard entry of symptoms and findings
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts, bins	Jointed arm and hand	Camera, joint angle sensors
Refinery controller	Purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, flow, chemical sensors
English tutor	Student's score on test	Set of students, testing agency	Display of exercises, corrections	Keyboard entry, student responses

Note how differently "performance" is defined in each row, and how much each definition leaves out.

Examples of Agents

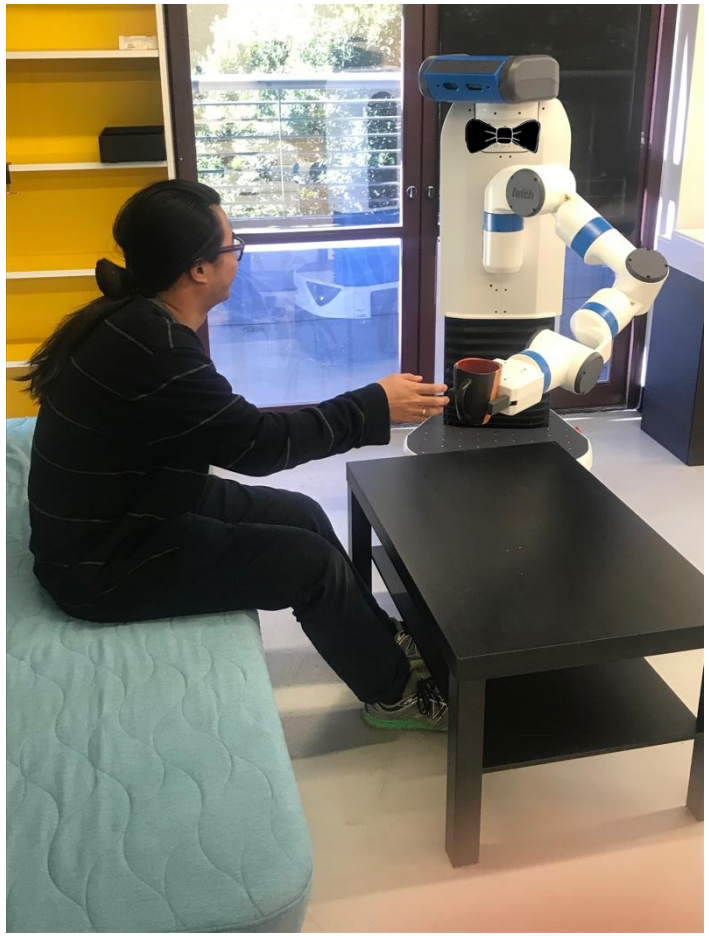
Shopping Agent

- Actuators
  - *Follow URL, fill in form, ask user for information (credit card, preferences, address)*
- Sensors
  - *IP packet scanning tools, Information retrieval algorithms run on news, product webpages*
- Environment: everything relevant to the agent's decision-making
  - *WWW sites, vendors + repositories, user (and their current activity)*
- Performance measure: a function of the sequence of environment states induced
  - *Some combination of: price; appropriateness; time taken to place the order; time taken for the item to arrive; quality of the item*

Image: https://www.morethanshipping.com/internet-things-iot-will-help-logistics/

Household Robot

- Actuators
  - Joint motors, wifi antennas
- Sensors
  - RGBD cameras, LiDAR, joint values, mic
  - Object detection algorithms, localization algorithms, user-command interface
- Environment
  - Location of humans, locations of all objects in the household, lighting conditions
- Performance measure
  - Number of questions asked, time taken to serve, cost of resources used



In-Class Exercise: Write the PEAS

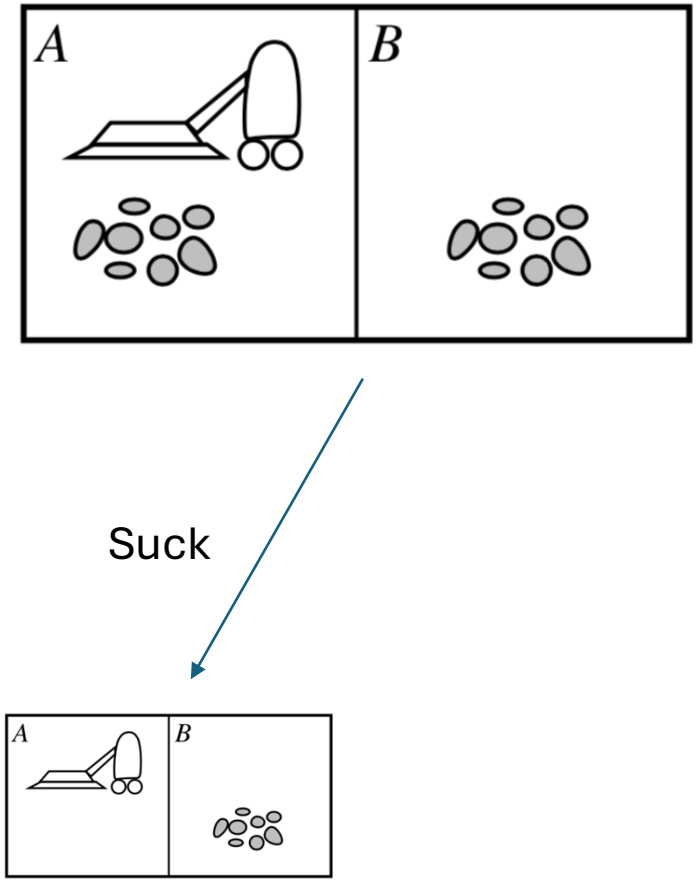
- Pick **one** of these and write its full PEAS specification. You have three minutes.
  - **(a)** An agent that plays chess against a human opponent
  - **(b)** An autonomous drone inspecting high-voltage transmission lines
  - **(c)** A spam filter for the institute mail server
  - **(d)** An agent that allocates lecture rooms across the department
- Then answer three questions about what you wrote:
  - Which component of **P** was hardest to state precisely?
  - Where do the parts of **P** conflict with one another?
  - What does your agent **fail to perceive** that it would need in order to score well?
- The third question is usually the one that kills a design.

Specifying Environments

Environments

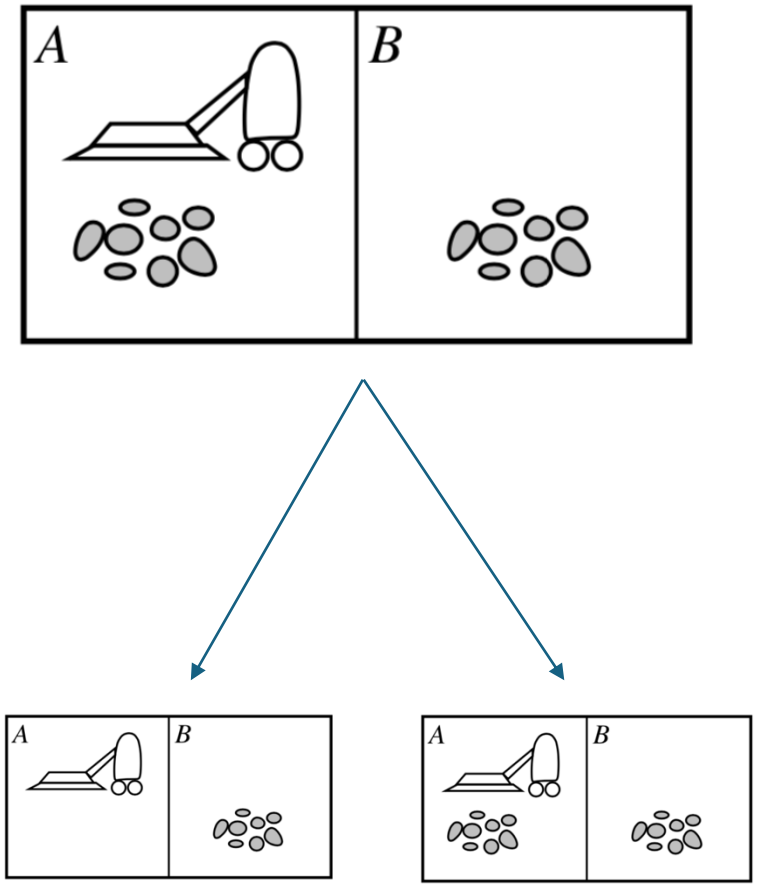
- A property satisfies the Markov assumption iff its value is independent of the past given the present
  - E.g., Markovian action model: next state depends only on current state, action
  - Bounded histories can be expressed as a part of the state

- Environment models are designed to make actions Markovian



Environments

- A property satisfies the Markov assumption iff its value is independent of the past given the present
  - E.g., Markovian action model: next state depends only on current state, action
  - Bounded histories can be expressed as a part of the state
- Environment models are designed to make actions Markovian



Can you think of a problem where the environment or the utility function is not Markovian?

Types of Environments

	Solitaire	Chess	Internet Shopping	Household Robot
Observable				
Deterministic				
Static				
Discrete				
Single Agent				

- Observable: Agent knows the state of environment at all times
- Deterministic: Effects of actions are expressible as functions: $S \times A \rightarrow S$
- Static: Environment does not change if agent does not act
- Discrete: Environment modeled using discrete variables, discrete timesteps
- Single agent: Problem includes a single decision-maker

Types of Environments

	Solitaire	Chess	Internet Shopping	Household Robot
Observable	No	Yes	Partial	Partial
Deterministic	Yes	Yes*	No	No
Static	Yes	No*	No	No
Discrete	Yes	Yes	Yes	No
Single Agent	Yes	No*	No [Communication?]	Yes

- Observable: Agent knows the state of environment at all times
- Deterministic: Effects of actions are expressible as functions: $S \times A \rightarrow S$
- Static: Environment does not change if agent does not act
- Discrete: Environment modeled using discrete variables, discrete timesteps
- Single agent: Problem includes a single decision-maker

Episodic vs. Sequential Environments

- **Episodic**: experience divides into atomic, independent episodes
  - Each episode is one percept and one action
  - The action taken in one episode does not affect any later episode
  - Spotting defective parts on an assembly line; classifying an image

- **Sequential**: the current decision affects all future decisions
  - Chess, driving, robot navigation, a conversation
  - Short-term actions have long-term consequences, so the agent must think ahead

- Why this axis matters more than it looks
  - Episodic problems need no lookahead at all; they are enormously simpler
  - This is the difference between classification and planning

- **Common confusion:** episodic is about **dependence between episodes**, not about whether the task repeats. A part-picking robot repeats its task forever and is still episodic.

Deterministic, Stochastic, and Nondeterministic

- **Deterministic:** the next state is completely determined by the current state and the action executed

- **Stochastic:** outcomes are uncertain, and the uncertainty is **quantified by probabilities**
  - The wheels slip 10% of the time; the sensor is wrong 2% of the time
  - The agent can reason about likelihoods and maximise expected utility

- **Nondeterministic:** outcomes are listed as possibilities, with **no probabilities attached**
  - A plan must succeed for **every** possible outcome, not merely the likely ones
  - This is the right model when probabilities are unknown or when you need a worst-case guarantee

- An environment is **uncertain** if it is not fully observable **or** not deterministic
  - This distinction decides the **form of the solution**: a fixed sequence, a contingent plan, or a policy, which is exactly where this lecture is headed

Known vs. Unknown Is Not the Same as Observable

- **Known** / **unknown** refers to the agent's knowledge of the **rules**: the transition model
- **Observable** / **partially observable** refers to what the **sensors** report about the current state
  - These are different questions, and all four combinations occur

- **Known + fully observable**
  - Solitaire with the rules in front of you
- **Known + partially observable**
  - A card game whose rules you know, but whose opponent's hand you cannot see
- **Unknown + fully observable**
  - A new video game: the screen shows you everything, but you do not yet know what the buttons do
- **Unknown + partially observable**
  - A robot switched on in an unmapped building

- Unknown environments force the agent to **explore** in order to learn the model.

Types of Environments: All Seven Axes

	Solitaire	Chess (clock)	Internet Shopping	Household Robot	Part-Picking Robot
Fully observable	Yes	Yes	Partial	Partial	Partial
Deterministic	Yes	Yes	No	No	No
Episodic	No	No	No	No	Yes
Static	Yes	Semi	No	No	No
Discrete	Yes	Yes	Yes	No	No
Single agent	Yes	No	No	Yes	Yes
Known	Yes	Yes	Partly	No	Partly

The two rows your earlier table did not have are Episodic and Known, and they are the two that most often decide which algorithm applies.

In-Class Exercise: Classify the Environment

- For **each** of the following, fill in all seven axes:
  - **(a)** A self-driving car in Chennai traffic
  - **(b)** A poker bot playing six-handed online
  - **(c)** A medical diagnosis system in a hospital
  - **(d)** An agent that trades equities on the NSE
- Discussion questions:
  - Which axis did your group disagree about most? Why?
  - Pick one agent and **give it better sensors**. Which entries change?
  - Pick one and **remove a sensor**. Does the problem become harder, or merely different?
- **The point:** these labels are properties of the task environment as you have specified it, not facts about the world. Changing the specification changes the answers.