August 26: Agents

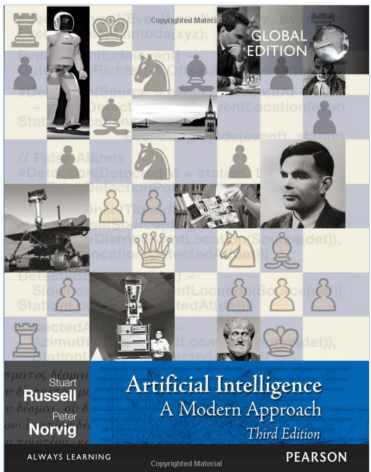
Opening Example: Which of the following are in violation of the CU Academic Integrity Policy?

1. For an assignment, Chris searches the internet for relevant codes and copy-pastes them into his Jupyter Notebook. He properly cites the source of the codes.

2. Example 2: For an assignment, Maciej and Felix work together to figure out how to implement the codes, but each works on their own computer and develops their own software.

3. Example 3: For an assignment, Rhonda has a plan for how to implement an algorithm, but isn't sure how to manipulate a Python list in a particular way that she needs to. She searches the internet, finds a fix, and implements it in her code without copying it.

Text



Artificial Intelligence: a Modern Approach (Third edition)
by Russell and Norvig
International, older, and PDF editions will work: just make sure to match any section numbers that changed.
Additional reading will be linked to the course calendar as needed

Announcements and To-Dos

Announcements:

1. Syllabus Grading Scale was correct:
   (35%) Homework assignments total (every 1-2 weeks, lowest dropped, late days)
   (15%) *Each* of 2 Exams
   (15%) *Each* of 2 Practicums
   (5%) Participation (Canvas) (start after Friday)
2. Zach's Office Hours for this week: Tu/Th 10a-12p; F: 1p-1:40p

Last time we learned:

1. Nothing really, but we had fun.

To do:

1. Make sure you can access the Canvas page and read the syllabus
2. Install Anaconda (or other reliable Jupyter notebook method)

What is an Agent?

Definition: An *agent* is an entity that perceives and acts.

- It perceives via sensors, yielding *percepts*
- It acts via actuators, performing *actions*

What is an Agent?

Definition: An *agent* is an entity that perceives and acts.

- It perceives via sensors, yielding *percepts*
- It acts via actuators, performing *actions*

Definition: A *percept* is the agents perceptual inputs at any given instant.
Definition: A *percept sequence* is the complete history of everything the agent has ever perceived.

What is an Agent?

Definition: An *agent* is an entity that perceives and acts.

- It perceives via sensors, yielding *percepts*
- It acts via actuators, performing *actions*

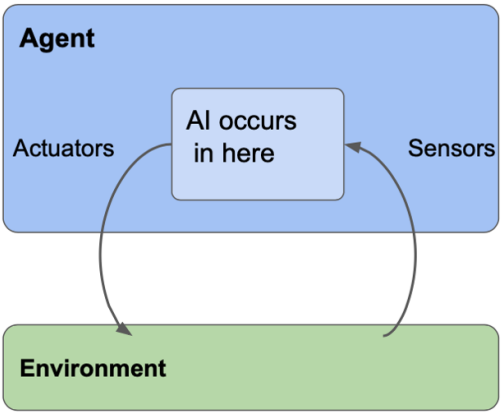
Definition: A *percept* is the agents perceptual inputs at any given instant.
Definition: A *percept sequence* is the complete history of everything the agent has ever perceived.

Definition: An *agent function* maps the percept sequence to an action. It is implemented by an *agent program*.

Definition: A *rational agent* is one that does the "right" thing.

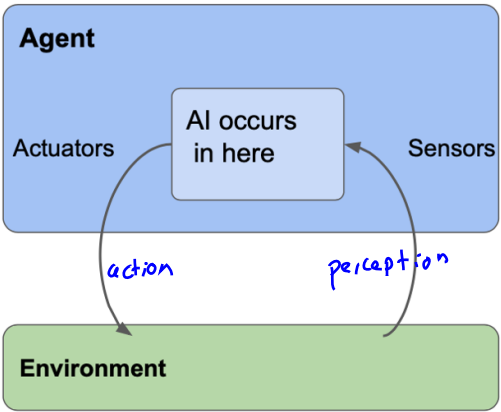
Perception and Action

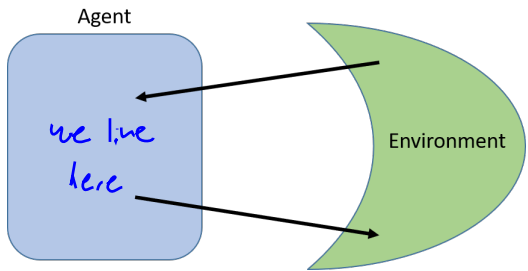
The agent is often broken down into a Perception-Action cycle:



Perception and Action

The agent is often broken down into a Perception-Action cycle:



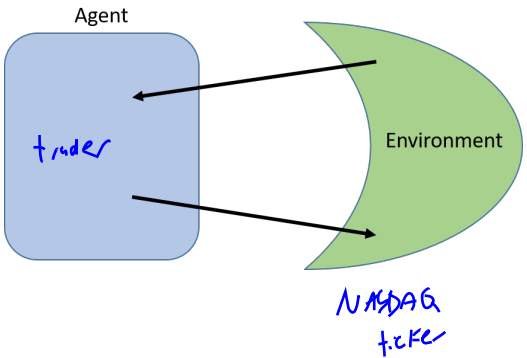


The simpler breakdown.

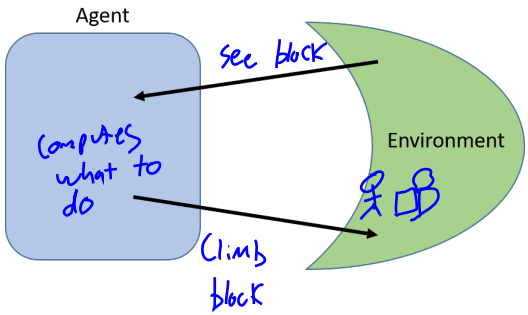
Agents

In this class, we work on the function(s) that drive the agents' behavior. You generally can't control the environment, but you can control the agent!

Example: Trading Agents

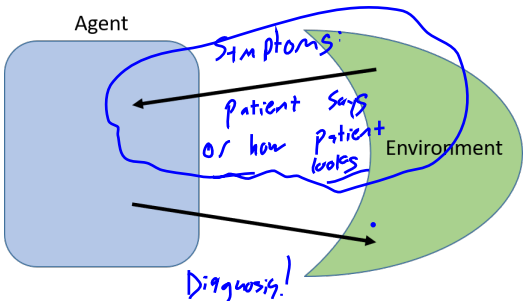


Example: Robotics

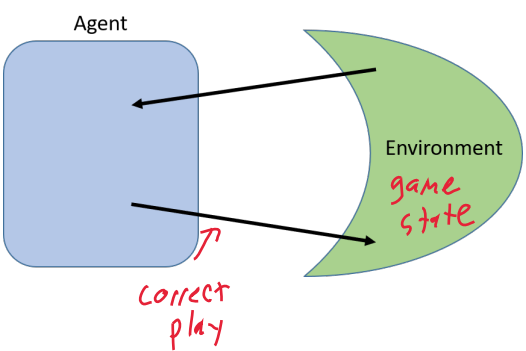


Agents

Example: Medicine



Example: Games



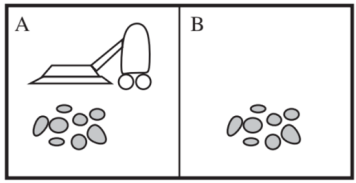


Figure 2.2 A vacuum-cleaner world with just two locations.

Percept sequence	Action
$[A, Clean]$	*Right*
$[A, Dirty]$	*Suck*
$[B, Clean]$	*Left*
$[B, Dirty]$	*Suck*
$[A, Clean], [A, Clean]$	*Right*
$[A, Clean], [A, Dirty]$	*Suck*
⋮	⋮
$[A, Clean], [A, Clean], [A, Clean]$	*Right*
$[A, Clean], [A, Clean], [A, Dirty]$	*Suck*
⋮	⋮

Figure 2.3 Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure 2.2.

Vacuum World

Performance Measures

Definition: The *performance measure* evaluates the behavior of the agent in an environment. A rational agent acts so as to maximize the expected value of the performance measure given the precept sequence it has seen so far.

Example: Vacuum Cleaner World

1. We *could* measure performance by counting the amount of dirt cleaned up in a single eight-hour shift.

Performance Measures

Definition: The *performance measure* evaluates the behavior of the agent in an environment. A rational agent acts so as to maximize the expected value of the performance measure given the precept sequence it has seen so far.

Example: Vacuum Cleaner World

1. We *could* measure performance by counting the amount of dirt cleaned up in a single eight-hour shift.
2. We *could* reward the agent for the floor being clean. One point per clean square, with a penalty for electricity usage.

Performance Measures

Definition: The *performance measure* evaluates the behavior of the agent in an environment. A rational agent acts so as to maximize the expected value of the performance measure given the precept sequence it has seen so far.

Example: Vacuum Cleaner World

1. We *could* measure performance by counting the amount of dirt cleaned up in a single eight-hour shift.
2. We *could* reward the agent for the floor being clean. One point per clean square, with a penalty for electricity usage.
3. WWYD? **ZOOM POLL O'CLOCK**

Rationality

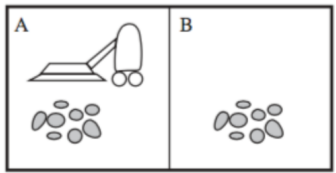
Our task in designing an AI is to design performance measures to track what we actually want in the environment.

One Vacuum cleaner world agent might be programmed such that:

1. If the current square is dirty, vacuum it.
2. If the current square is clean, move to the other square.

Is this agent **rational?** ✓

Vacuum World



Rationality

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

Rationality is based on the following 4 principles:

1. A performance measure (utility)
2. environment familiarity
3. actions that are possible
4. sequences of percepts (memory)

The PEAS

Those 4 principles are often broken down into the *task environment,* which is essentially the "problems" that our rational agents are the "solution" to.

PEAS: **P**erformance, **E**nvironment, **A**ctuators, **S**ensors

Example: a PEAS description of the task environment for a taxi:

Agent Type	**Performance Measure**	**Environment**	**Actuators**	**Sensors**
Taxi Driver	safe, fast, legal passenger comfort maximize profit	roads, other cars, pedestrians customers	steering, accelerator, brakes, turn signals, horn, display	cameras, sonar speedometer, GPS engine sensors, keyboard

Observability

Environments are often classified according to their dominant properties.

Fully Observable	vs.	**Partially Observable**	vs.	**Unobservable**
Agent can observe all relevant aspects of the environment		Some parts missing from sensor data		Must make a choice with no observations
No need to maintain memory to keep track.		Needs memory by agent to make optimal decision.		
Current information is sufficient for decision		e.g. noisy environment or inaccurate sensors		

Randomness

Deterministic	vs.	**Stochastic**
No randomness when you move a piece		Can't predict outcome of the dice
Effect of moving a piece is complete predetermined.		There is randomness present in the environment.

Action Cardinality

Discrete	vs.	**Continuous**
Finitely (or countably) many actions		Uncountably infinite many actions.

Environment and Competition

Benign	vs.	**Adversarial**
No other objectives might contradict your own		An opponent is out to get you or has exclusive goals

Single Agent	vs.	**Multi-Agent**

Timelines

Episodic	vs.	**Sequential**
Agents experience *episodes:* Percept $\rightarrow$ single action Next episodes is not dependent on prior actions		Current decisions may affect all future decisions

Static	vs.	**Dynamic**
The environment remains constant.		The environment can change while the agent is deliberating

Example Environment

Consider the game of **Chess:**

Is this environment:

Partially	or	Fully Observable
Deterministic	or	Stochastic
Continuous	or	Discrete
Benign	or	Adversarial

Example Environment

Consider the game of **Poker:** (say, Texas Hold'em)

Is this environment:

Partially	or	Fully	Observable
Deterministic	or		Stochastic
Continuous	or		Discrete
Benign	or		Adversarial

→ cards / outcomes

how much $

- show bluff

Example Environment

Consider the game of **Super Mario:**

Is this environment:

✓ Partially	or	Fully Observable
Deterministic	or	Stochastic ✓
✓ Continuous	or	Discrete
Benign	or	Adversarial

goal decides which problem we solve

Example Environment

Consider designing a robotic/autonomous car.

Is this environment:

Partially	or	Fully	Observable
Deterministic	or		Stochastic
Continuous	or		Discrete
Benign	or		Adversarial

Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Backgammon	Fully	Multi	Stochastic	Sequential	Static	Discrete
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic	Continuous
Medical diagnosis	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continuous
Part-picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continuous
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Interactive English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

Figure 2.6 Examples of task environments and their characteristics.

Agent Types

There are a few types of agents - algorithms! - that we'll explore in this class, including:

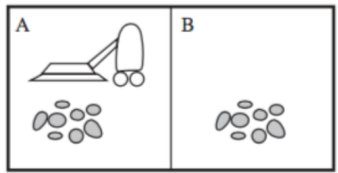
1. Simple reflex agents
2. Model-based reflex agents
3. Goal-based agents
4. Utility-based agents

These delineations are not cut and dried: agents can be of multiple types!

Simple Reflex Agent

Vacuum World:

```
In [ ]: def reflexVacuum(location, status):
            if status=='dirty':
                # clean it
            elif location==1:
                # move right
            elif location==2:
                # move left
```



- Select an action based on the *present*
- Do not consider future consequences
- Encode preferences as a function of the percepts and actions.

Model-based Reflex Agent

Vacuum World:

In [ ]:
```
def updateState(percept, status_old):
    # update agent's idea of what the world looks like
    return status

def ruleMatch(status, rules):
    # match which action (reflex) should follow the given state
    # according to the rules
    return action

def reflexVacuum(location, percept, status_old):
    status = updateState(percept, status_old)
    action = ruleMatch(status, rules)
```

- Select an action based on the *past and present*
- Still not thinking about future consequences ("*reflex*")
- Maintains an internal state to keep track of what the world looks like

Other Agents

A *goal-based agent*:

- May select actions based on part, present, and future
- Works towards a pre-specified goal
- Decisions based on hypothesized consequences - requires a model!

A *utility-based agent*:

- May select actions based on part, present, and future
- As opposed to a set goal, quantifies and scores benefits of various actions.

A *learning agent*:

- Idea: don't build an intelligent machine, build a learning machine and then **teach** it.
- Separate percepts into rewards and penalties
- Example: spam filters

Moving Forward

- Before next class:
  1. Make sure you can access the Canvas page and read the syllabus
  2. Install Anaconda (or other reliable Jupyter notebook method)
  3. If new to Jupyter/Python3: Review and complete Numpy/Pandas tutorials if needed
  4. Download the in-class notebook! (posted by tomorrow am).
- Next time: **Agents** and **States**.