CS2109S: Introduction to AI and Machine Learning

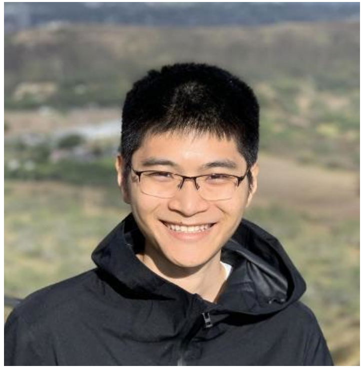
Lecture 1: Intro to CS2109S & Artificial Intelligence

12 Aug 2022

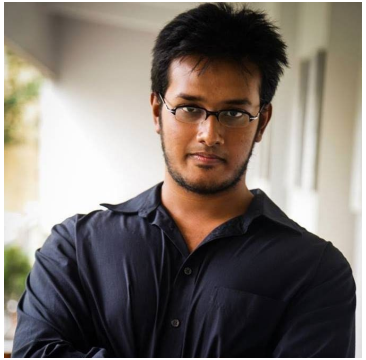

acebook

Rizki

Jonathan

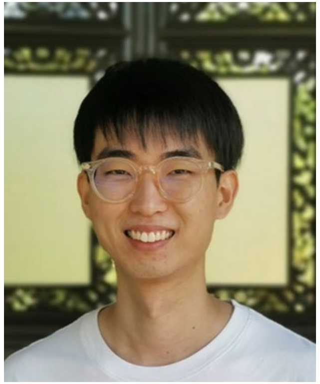


Zitai



Rafeed

Kuluhan



Soo Han

Austen

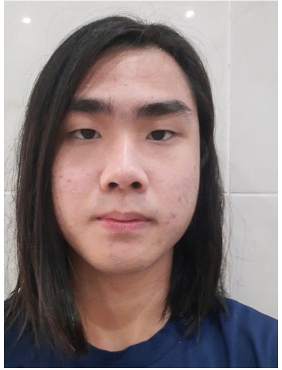
Yuming

Jin Feng

Ming Chong

Aloysius

Rahul

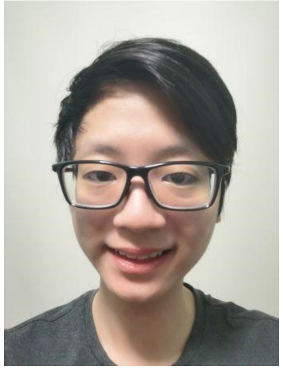


Audrey

Arnav

Nicholas

Nyamdavaa



Gabriel

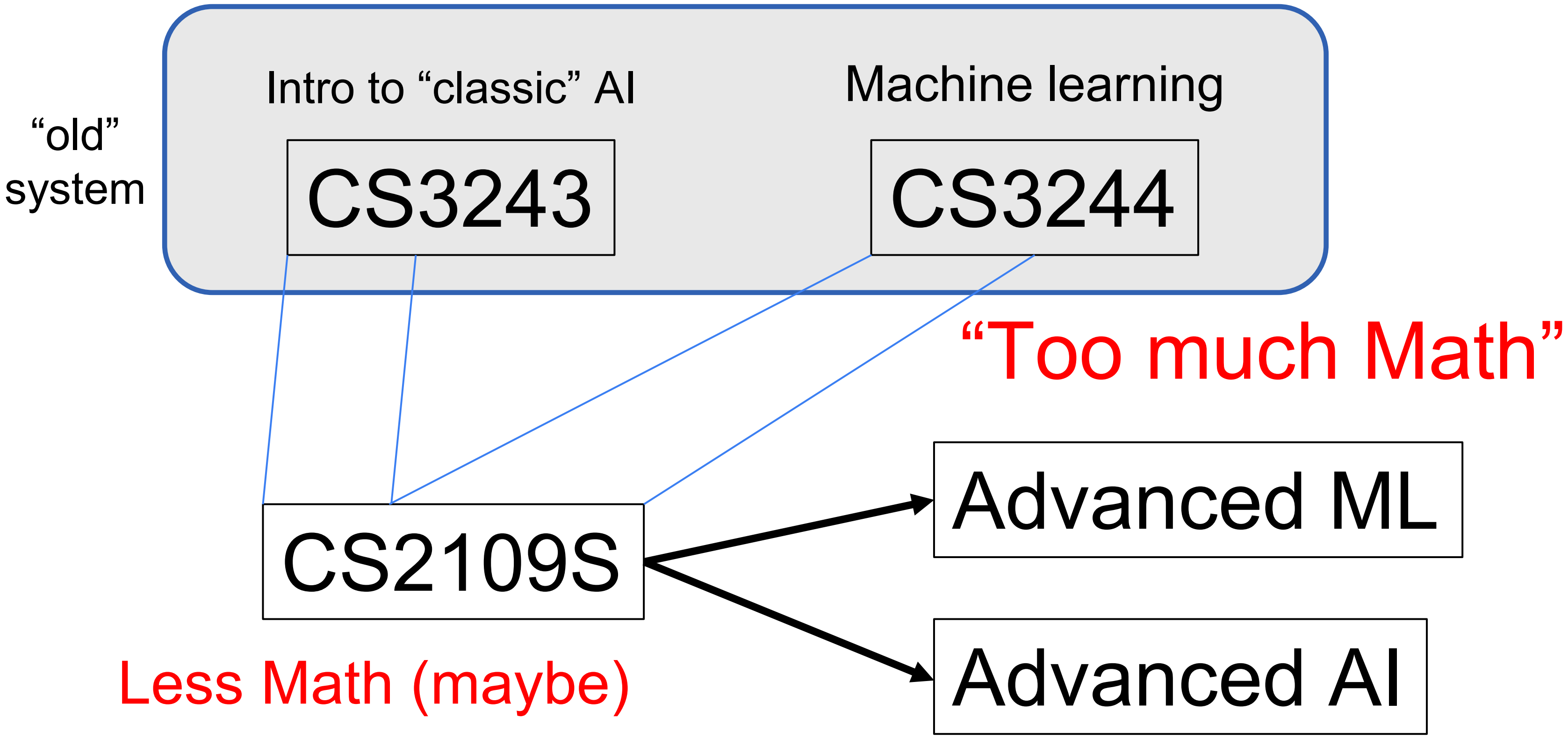
Rights Infringements on NUS Course Materials

All course participants (including permitted guest students) who have access to the course materials on LumiNUS or any approved platforms by NUS for delivery of NUS modules are not allowed to re-distribute the contents in any forms to third parties without the explicit consent from the module instructors or authorized NUS officials

Why are you here?

Why are we here

- We were teaching the same things I taught in 2006/7 in CS3243
- Machine/Deep Learning is a big deal.
- CS2109S is now compulsory for all CS students
- Some CS majors might only take one AI class before graduation.



Less Math (maybe)

More application (coding)

Focus on key ideas & intuitions

Course Pre-Requisites

- CS1101S, CS1010S or equivalent
- CS1231 or equivalent
- MA1521 (calculus)
- CS2040S or equivalent
- Linear algebra (some)
- Python

Problem Set 0

language-agnostic

Warning: we are *still* fine tuning this class.

Topics

1. Search (“classic AI”)
2. Decision Trees
3. Linear Regression
4. Gradient descent
5. Logistic Regression

Topics

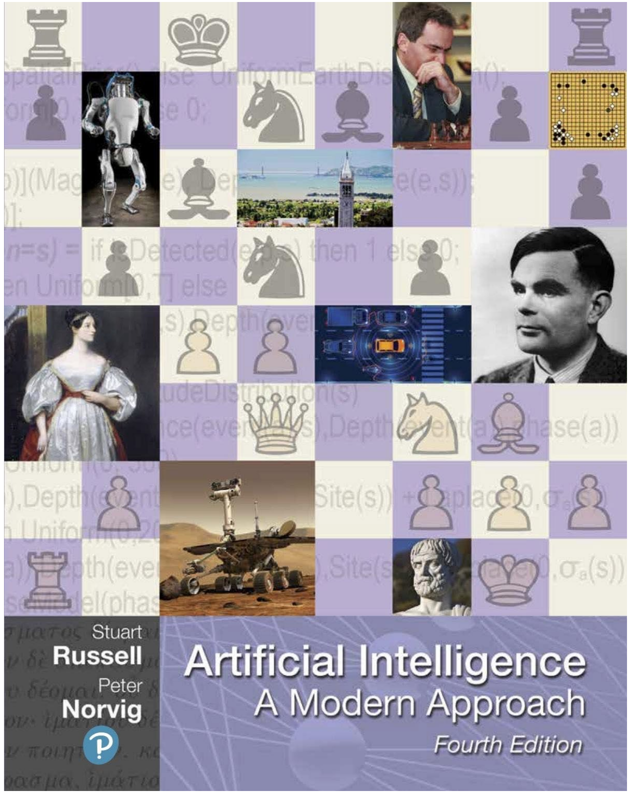
6. Support Vector Machines
7. Artificial Neural networks
8. Unsupervised learning
9. AI & Ethics

Textbook

Russell and Norvig (2021) Artificial Intelligence: A Modern Approach (4th Ed)

AIMA

No need to buy!



Don’t memorize anything

Focus on concepts & understanding!

There will be webcast

But I don’t control when it appears on Luminus

IMPORTANT

Tutorial scheduling survey

Due 14 Aug 23:59!

Plagiarism Policy

- Okay to discuss, cite them
- Do not take away any code from your discussions.
- We will run plagiarism checker

Plagiarism Policy

- Do not use code from seniors
- Do not publish code on GitHub!!
- We will run plagiarism checker against assignments from previous batches

Learn how to learn

Learn to solve problems

Homework needs to be done to learn

Shouldn’t be high stakes

Plagiarism will be a problem

Homework shouldn't be high stakes

Students need a fair grade

So how?

Grading

- Homework needs to be done to learn
- Plagiarism will be a problem
- Students need a fair grade

⇒Homework needs to be a high weightage

⇒Homework cannot have a high weightage

Life is hard. ☹

Grading

- Homework needs to be done to learn

⇒Homework needs to be a high weightage

- Plagiarism will be a problem

⇒Homework easy to get full marks

⇒Homework cannot have a high weightage

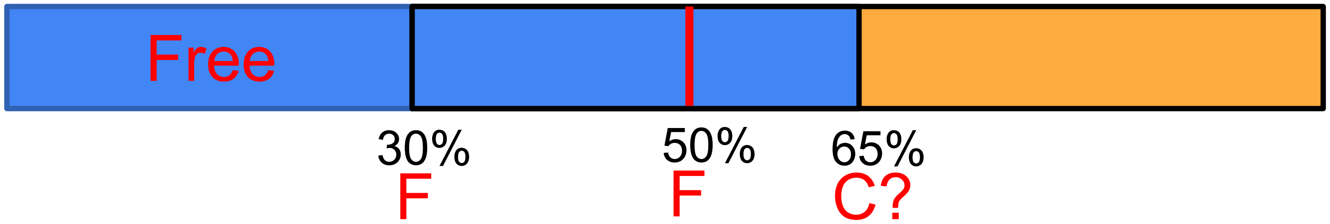
- Students need a fair grade

⇒ Exams mainly determine final grade

Grading

expected
to get 30%

- Coursemology (CA) – 30%
- Mini-Project – 10%
- Midterm Assessment – 30%
- Final (Practical) Exam – 30%



Coursemology, not Luminus/Canvas

But will use Luminus Gradebook to record marks!

Gamified CA (30%)

+100 EXP **+600 EXP** (every 2 weeks)

Lecture → Training → Tutorial → Problem Set

+50 EXP BONUS!

+2000 EXP

Mini-Project Contest **+1000 EXP bonus**

Please focus on learning, not grades

Gamified CA (30%)

Coding Practice
(every 2 weeks)

Turn up for class

Lecture → Training → Tutorial → Problem Set

Exploration

Mini-Project Contest

Enough EXP?

Level Up!

Final Level is CA grade

Background Survey

+100 EXP

Late Policy

- Up to 1 hour, no penalty
- Up to 24 hours, -20%
- Beyond 24 hours, -50%

Need extension, ask early

Assessment (Face-to-Face)

- Midterm Test (30%)
  - 30 Sep@10am (Fri, Week 7)
  - Venue: TBA

Might be evening because of venue

- Final Exam (30%)
  - 25 Nov@2.30 pm (Friday)

Arranging for Practical Exam

Exams

- Closed book, open-sheet (as good as open book)
- Focus on application, NOT memorization

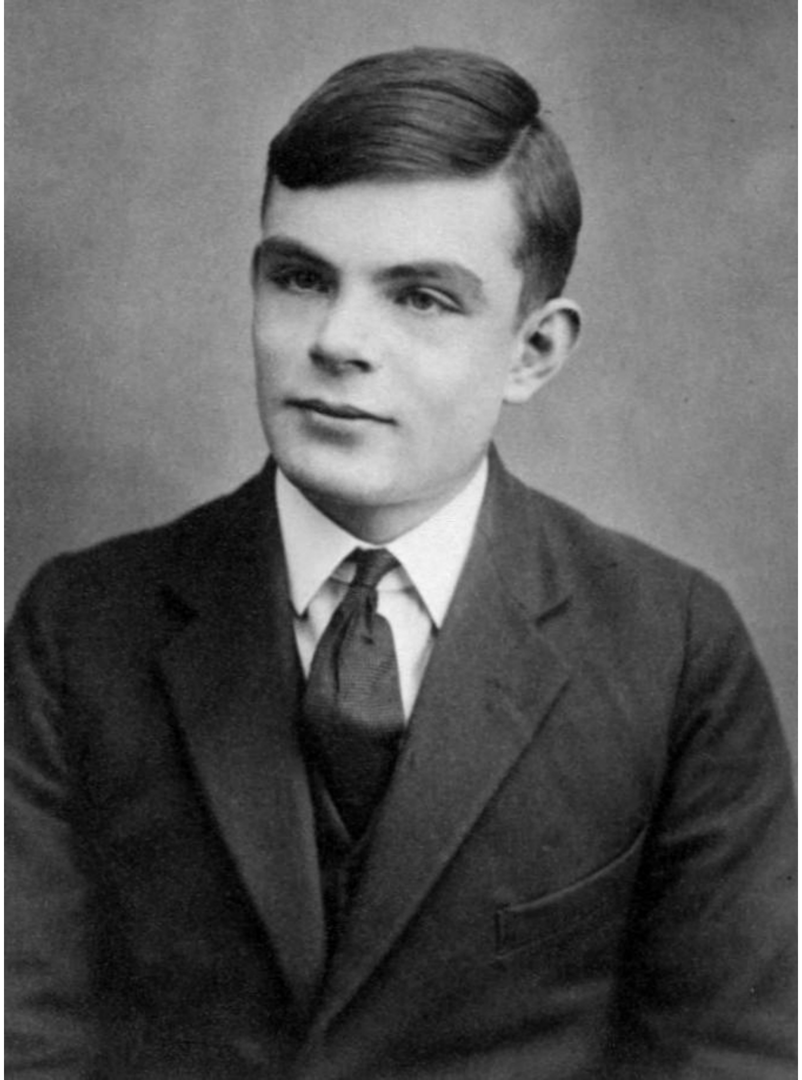
Questions?

Who wants a C for CS2109S?

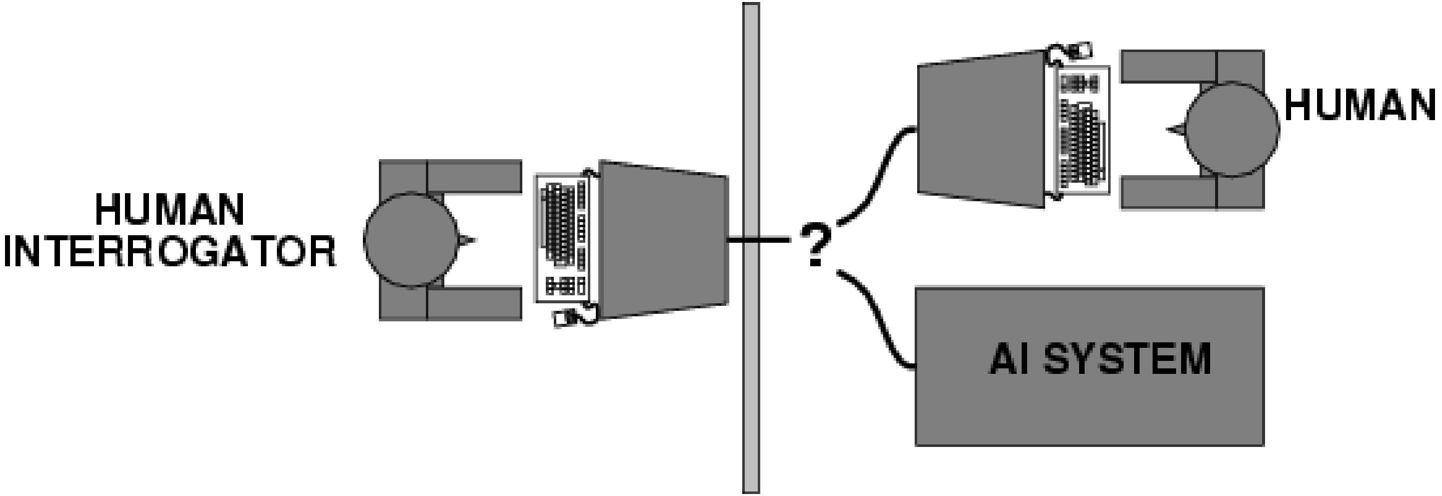
- Don’t procrastinate
- Start on homework and projects early
- Students who wait till the last minute to do the assignments invariably fail to debug their programs properly

Workload is front-loaded

What is AI?

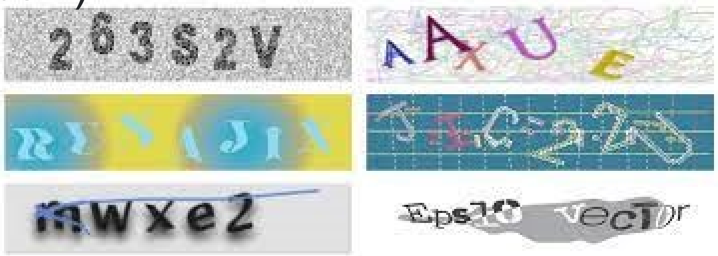

Credit: Wikipedia

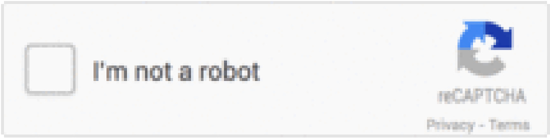
Turing Test (1950)

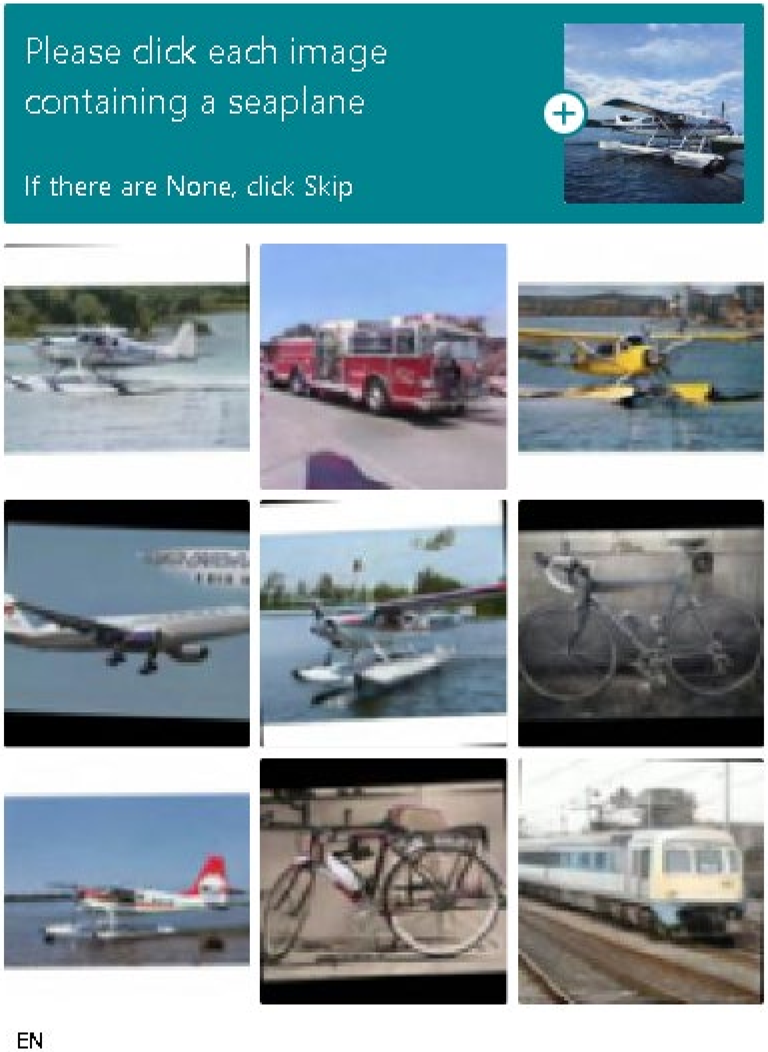


CAPTCHA

(Completely Automated Public Turing test to tell Computers and Humans Apart)







“Acting Humanly”

What else is considered AI?

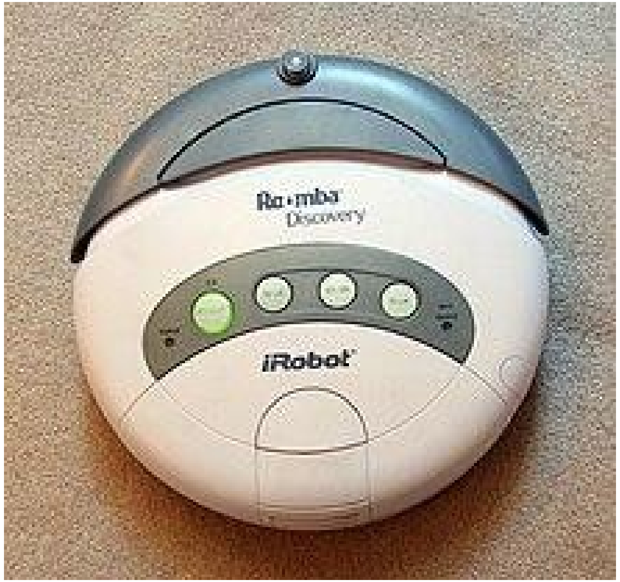
DeepBlue (1997)

Credit: IEEE Spectrum

AlphaGo (2015)

Credit: Guardian

iRobot Roomba (2002)



Credit: Wikipedia

Watson (2011) vs Jeopardy

Credit: NYTimes

Credit: Vulcan Post

Credit: Technology Review

Does it matter? ☺

Learning
how to learn

Learning
to ~~solve problems~~
frame problems

Approach AI as Problem Solving

Computer Science is the study of abstractions

Managing Complexity

Functional Abstraction

$$\text{Input} \longrightarrow \boxed{\text{Function}} \longrightarrow \text{Result}$$

Picture → $\boxed{\text{Function}}$ → Word/number

Picture → $\boxed{\text{Function}}$ → Chihuahua /muffin

Email → $\boxed{\text{Function}}$ → Spam /not spam

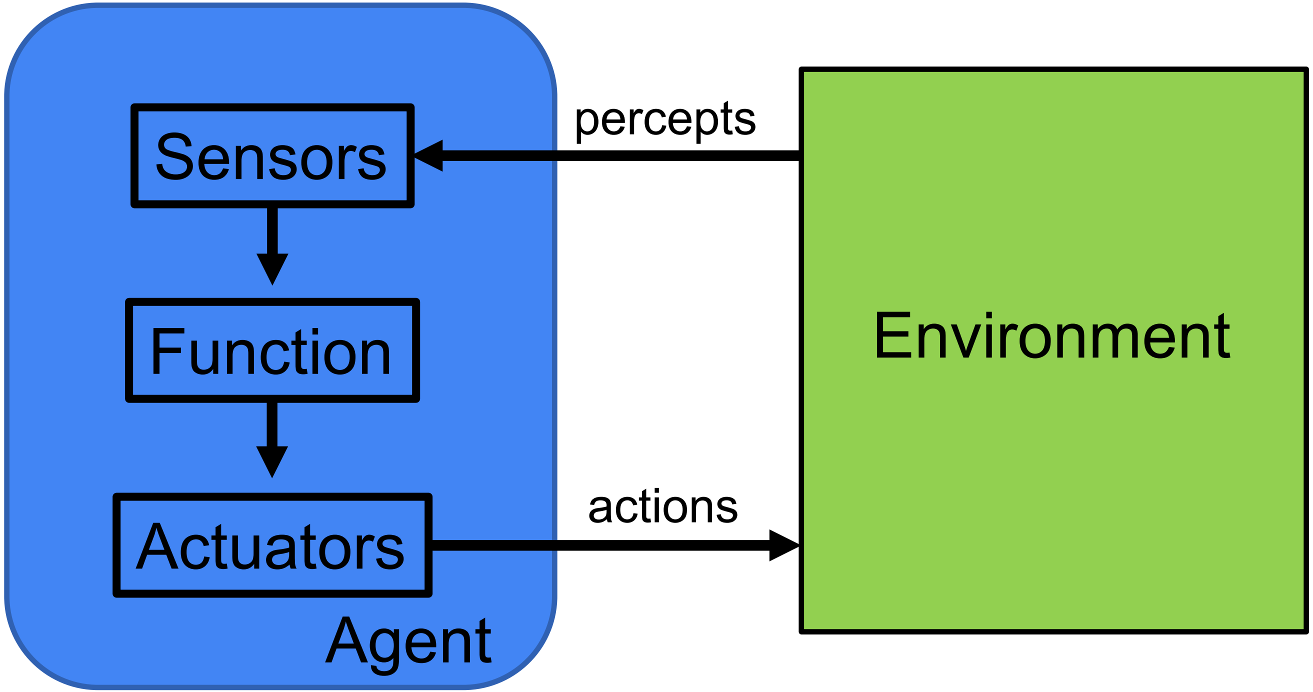
Black & white photo → $\boxed{\text{Function}}$ → Coloured photo

How do we write this function?

Is this sufficient?

How would we model self-driving?

Intelligent Agents



Agents

- An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators
- Human agent: eyes, ears, and other organs for sensors; hands, legs, mouth, and other body parts for actuators
- Robotic agent: cameras and infrared range finders for sensors; various motors for actuators

Agent design

The agent function maps from percept histories to actions:

$$[f: \mathcal{P}^{\star} \rightarrow \mathcal{A}]$$

The agent program runs on the physical architecture to produce *f*

agent = architecture + program

How would an agent know to do the right thing?

→need a measure of goodness

Performance measure

Performance Measure

1. Best for whom?
2. What are we optimising?
3. What information is available?
4. Unintended effects
5. What are the costs?

Performance vs. Cost

Rational agent

A rational agent will choose an action that is expected to maximize its performance measure, given the evidence provided by

- the percept sequence; and
- the built-in knowledge it has.

Rationality ≠ Omniscience

Agents can perform actions in order to modify future percepts so as to obtain useful information (information gathering, exploration)

An agent is autonomous if its behavior is determined by its own experience (with ability to learn and adapt)

Defining the Problem: PEAS

1. Performance measure
2. Environment
3. Actuators
4. Sensors

Autonomous Driving

Performance Measure

1. Safety
2. Speed
3. Legal
4. Comfort

Environment

1. Roads
2. Weather/visibility
3. Other vehicles
4. Pedestrians/obstacles

Actuators

1. Steering wheel
2. Accelerator
3. Brake
4. Signal
5. Horn

Sensors

1. Camera
2. LIDAR
3. Speedometer
4. GPS
5. Engine sensors

Medical Diagnosis System

Performance Measure

1. Health outcome
2. Cost
3. Lawsuits

Environment

1. Patient
2. Hospital (equipment, etc.)
3. Staff

Actuators

Screen display (questions,test, diagnoses, treatments, referrals)

Sensors

1. Keyboard
2. Medical Readings
3. Medical History

Interactive Coding Tutor

Performance Measure

Maximize student's score on test/Leetcode

Environment

1. Questions available
2. Language(s)

Actuators

Screen display (questions,corrections, suggestions)

Sensors

1. Keyboard
2. Input from database

Characterizing the environment

Fully observable (vs. partially observable): An agent's sensors give it access to the complete state of the environment at each point in time.

Deterministic (vs. stochastic): The next state of the environment is completely determined by the current state and the action executed by the agent. (If the environment is deterministic except for the actions of other agents, then the environment is strategic)

Characterizing the environment

Episodic (vs. sequential): The agent's experience is divided into atomic "episodes" (each episode consists of the agent perceiving and then performing a single action), and the choice of action in each episode depends only on the episode itself.

"Memoryless"

Static (vs. dynamic): The environment is unchanged while an agent is deliberating. (The environment is semi-dynamic if the environment itself does not change with the passage of time but the agent's performance score does)

Characterizing the environment

Discrete (vs. continuous): A limited number of distinct, clearly defined percepts and actions.

Single agent (vs. multi-agent): An agent operating by itself in an environment.

Environment types

	Chess with a clock	Chess without a clock	Autonomous driving
Fully observable	Yes	Yes	No
Deterministic	Strategic	Strategic	No
Episodic	Yes	Yes	No
Static	Semi	Yes	No
Discrete	Yes	Yes	No
Single agent	No	No	No

The environment type largely determines the agent design
The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent

Implementing Agents

An agent is completely specified by the agent function mapping percept sequences to actions

Goal: find a way to implement the rational agent function concisely

Playing Tic-Tac-Toe

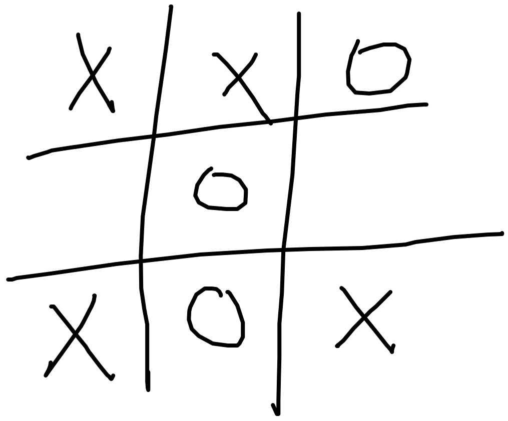


Table-lookup agent

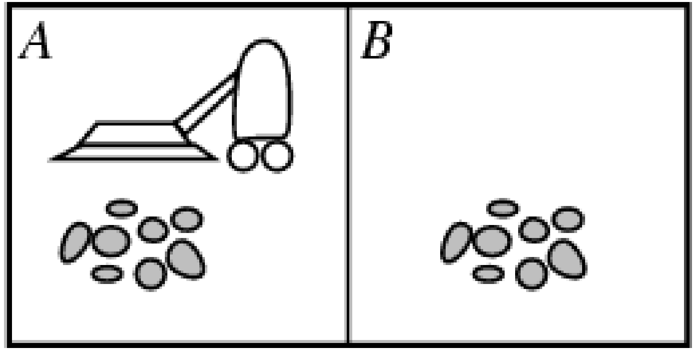
```
function TABLE-DRIVEN-AGENT(percept) returns action
    static: percepts, a sequence, initially empty
            table, a table of actions, indexed by percept sequences, fully specified

    append percept to the end of percepts
    action ← LOOKUP(percepts, table)
    return action
```

Possible drawbacks:

- Storage
- No autonomy/cannot react to changes
- Might need a long time to compute/learn the table entries

Vacuum-cleaner world



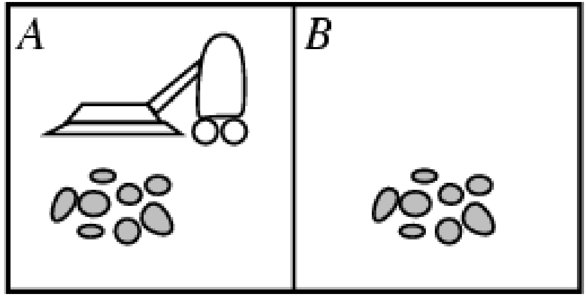
Percepts: location and contents, e.g., [A,Dirty]

Actions: *Left*, *Right*, *Suck*, *NoOp*

Vacuum Cleaner Agent

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*



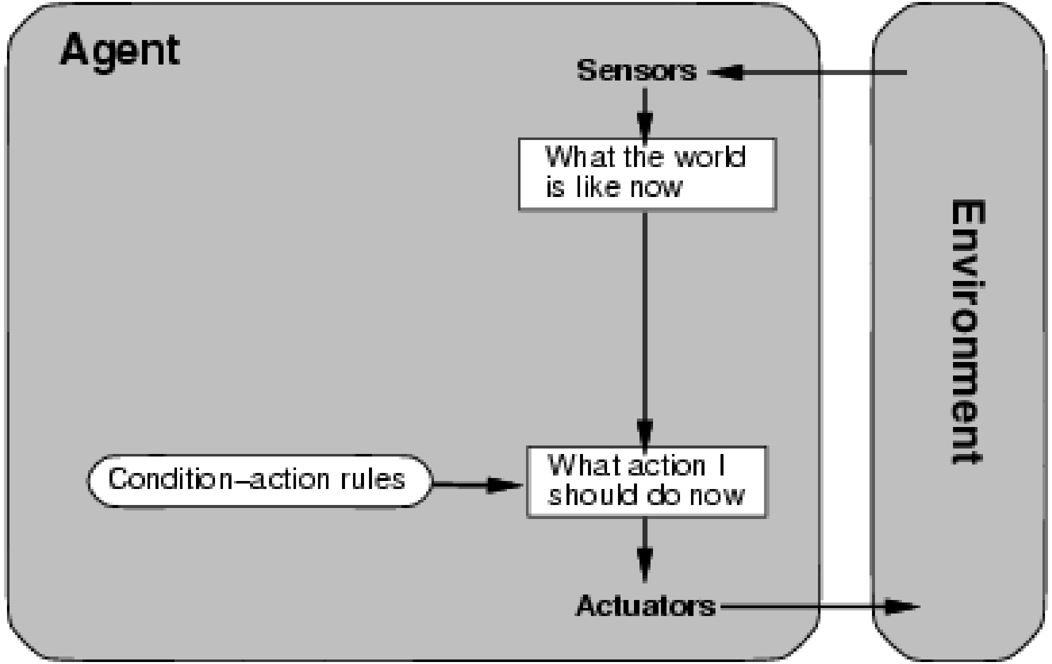
Lookup table as case statements

Models for Agent Organization

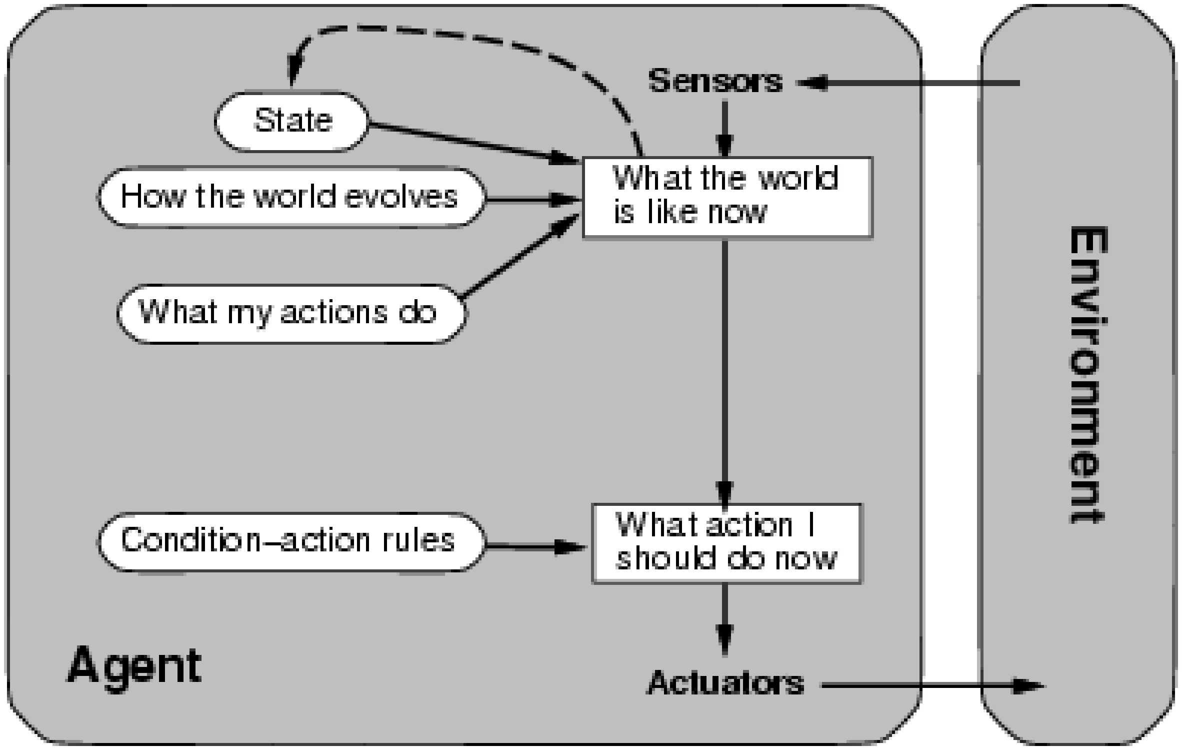
1. Simple reflex agents
2. Model-based reflex agents
3. Goal-based agents
4. Utility-based agents
5. Learning agents

Increasing complexity

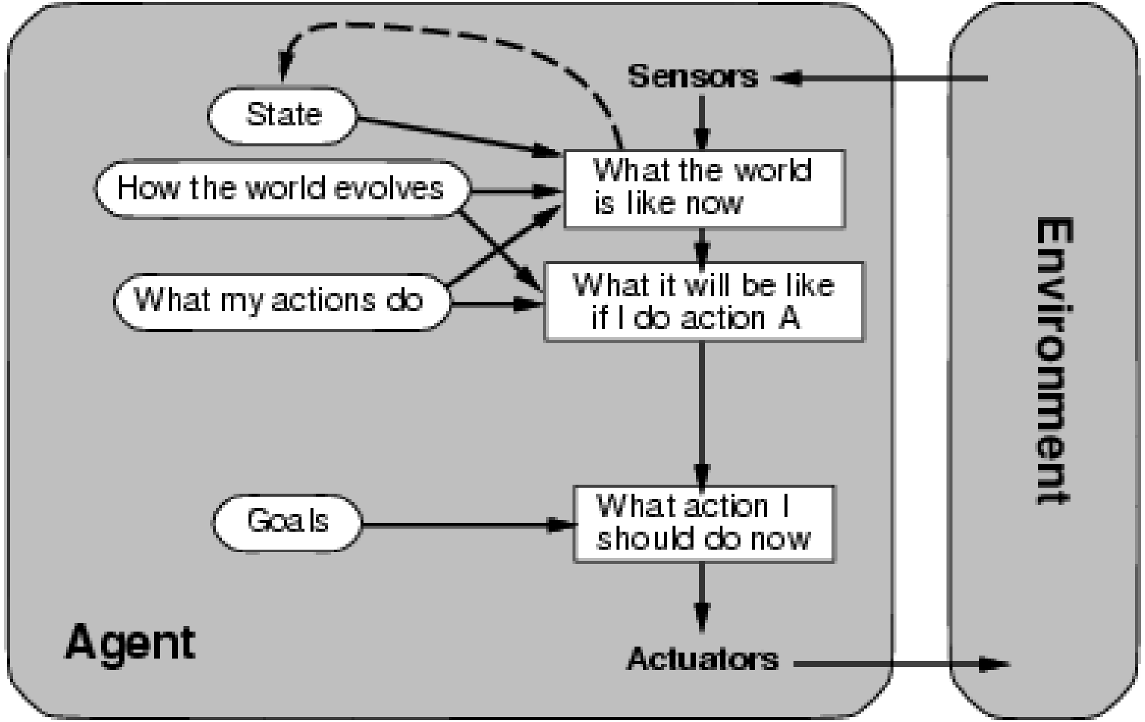
Simple Reflex Agent



Model-based Reflex Agent

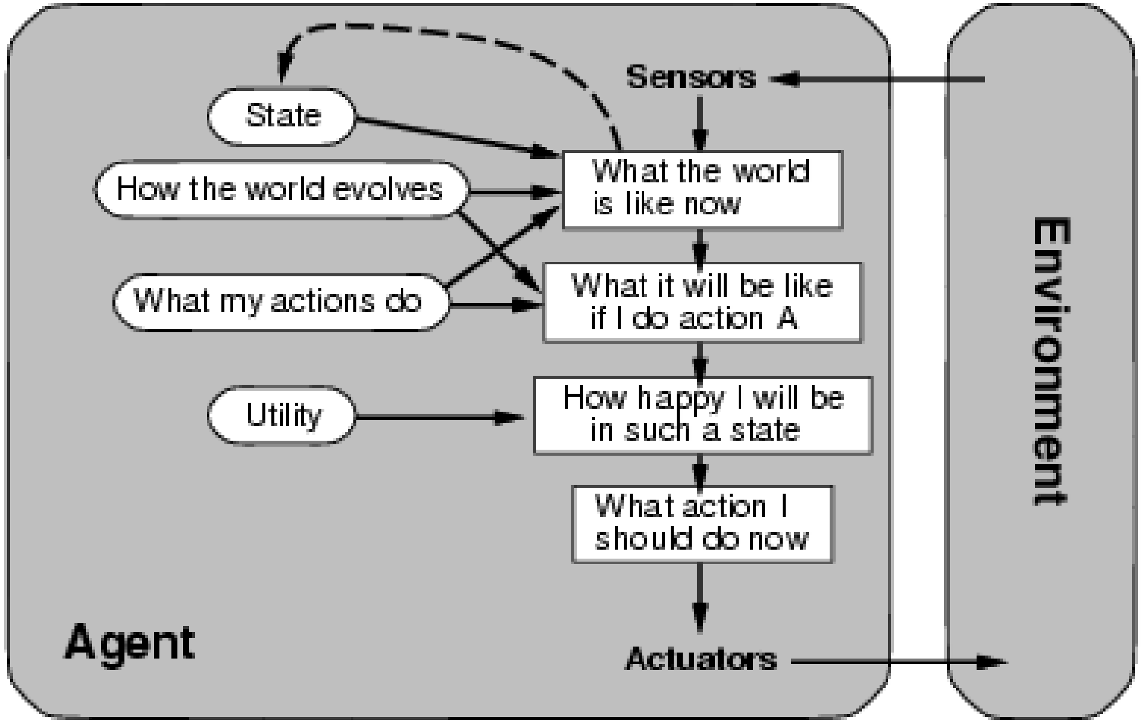


Goal-based Agent

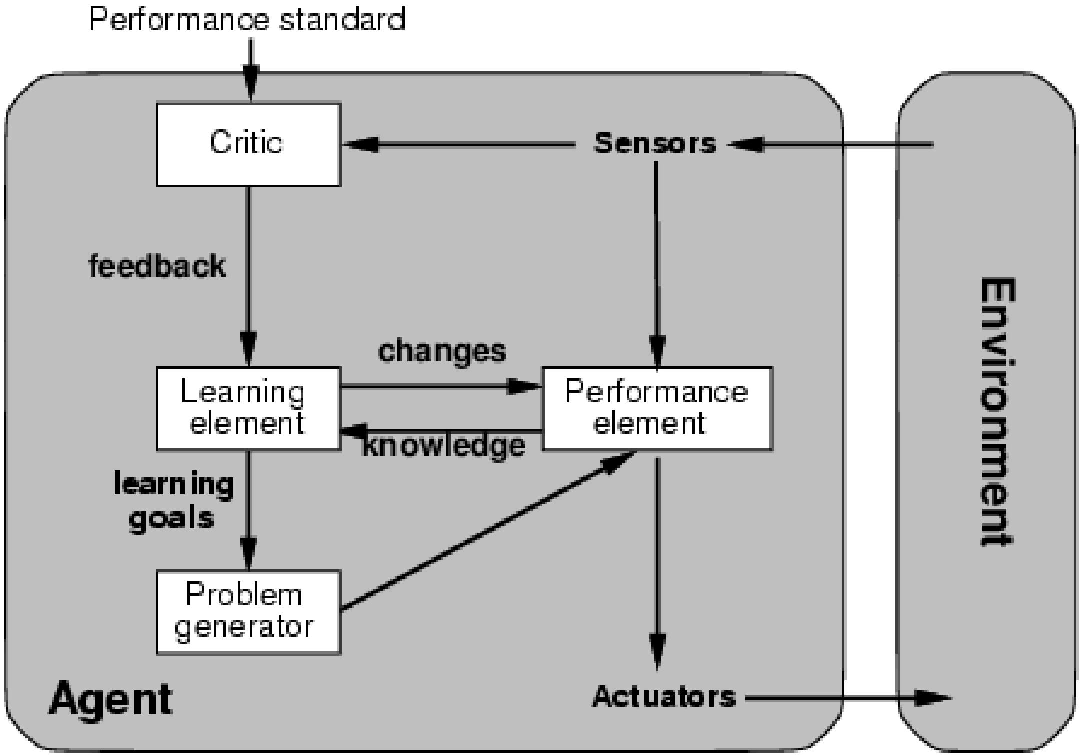


from Economics

Utility-based Agent



Learning Agent



A learning agent might not always maximize the performance measure!

There are no model answers in life.

Problem set solutions will not be released

Exploitation vs Exploration

An agent operating in the real world must often choose between:

- maximizing its expected utility according to its current knowledge about the world; and
- trying to learn more about the world

since this may improve its future gains.

This problem is known as the trade-off between ***exploitation*** and ***exploration***

Summary

- Frameworks to think about and understand general AI problems
- PEAS framework
  - Performance Measure: defines “good” solution in solution space
  - Environment: determines what we can and cannot do
  - Actuators: output
  - Sensors: inputs
- Typical Solutions (different classes of agents)
  - Implementing a lookup-table
  - Reflex, Model-based, Goal-based, Utility-based, Learning

Summary

- Why do we care about these models?

Condition	Algorithm	Time Complexity
No Negative Weight Cycles	Bellman-Ford Algorithm	$O(VE)$
On Unweighted Graph (or equal weights)	BFS	$O(V+E)$
No Negative Weights	Dijkstra’s Algorithm	$O((V+E)log\, V)$
On Tree	BFS / DFS	$O(V)$
On DAG	Topological Sort	$O(V+E)$

2040S anyone?

SSSP!

1. Tradeoff: Performance vs. Cost
2. Exploitation vs. Exploration

Do Lecture 1 Training!

Free +100 EXP

+50 EARLY BIRD BONUS

Please start on Problem Set 0 ASAP!

Due 20 Aug (Sat) 23:59