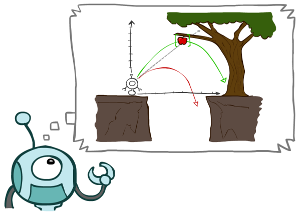
Artificial Intelligence

Module 2: Automated Problem Solving

PART 2.1: Intelligent Agent & Environment



(Slides adapted from StuartJ. Russell, B Ravindran, Mausam, Prof. Pallab Dasgupta, Prof. Partha Pratim Chakrabarti, Saikishor Jangiti

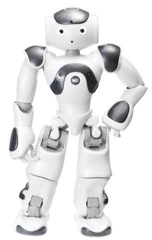
Module 2: Automated Problem Solving

- PART 2.1: Intelligent Agent & Environment
  - Agent : Intelligent agent
  - Rational Agents
  - Task environments: PEAS
  - Environment
  - Structure of Agents
- PART 2.2: Complex Problems and AI
- PART 2.3: Problem Solving Methods

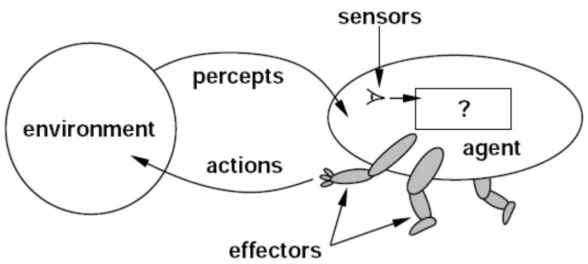
Identify

Intelligent Agents

- What is an agent ?
  - An agent is anything that can be viewed
    - as perceiving its environment through sensors and
    - acting upon that environment through actuators
  - Example:
    - **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
      - A thermostat detecting room temperature.



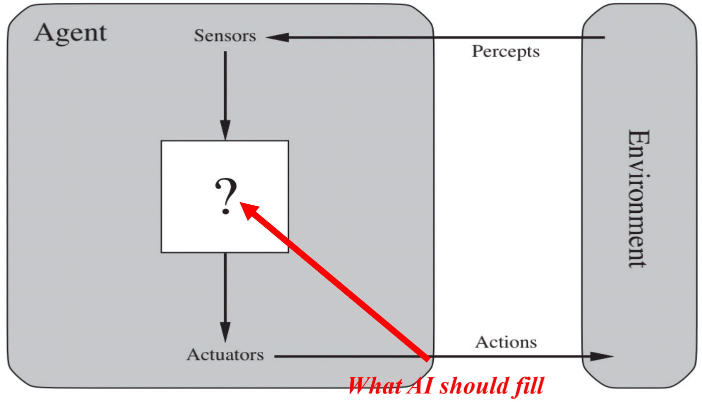
Intelligent Agents



- **Percept**
  - Agent's perceptual inputs at any given instant
- **Percept sequence**
  - Complete history of everything that the agent has ever perceived.

Agent's behavior is **mathematically** described by
Agent function maps from percept histories to actions:
$[f: \mathcal{P}^* \rightarrow \mathcal{A}]$
Agent program runs on the physical architecture to produce f

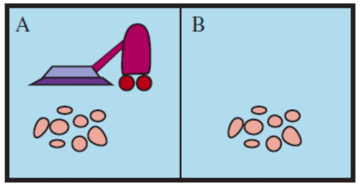
Diagram of an agent



Agents that Plan Ahead

An agent should strive to "do the right thing", *based on what it can perceive and the actions it can perform.*

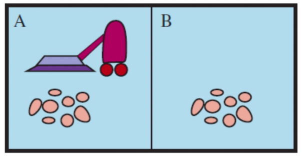
Example : Vacuum-cleaner world Agent



- Percepts:
  - location and contents(Clean or Dirty?),
  - e.g., [A , Dirty]
- Actions:
  - Left: Move left,
  - Right: Move right
  - Suck: suck up the dirt,
  - NoOp: do nothing

A vacuum-cleaner agent

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
⋮	⋮



Function Reflex-Vacuum-Agent([*location,status*]) return an action
If *status = Dirty* **then return** *Suck*
else if *location = A* **then return** *Right*
else if *location = B* **then return** *left*

Just tell me what to do

Rational agents

- What is rational at any given time depends on four things:
  - The performance measure that defines the criterion of success.
  - The agent's prior knowledge of the environment.
  - The actions that the agent can perform.
  - The agent's percept sequence to date.
- **DEFINITION OF A RATIONAL AGENT**
  - 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 percept sequence and whatever built-in knowledge the agent has.
  - Rationality vs omniscience

Example of a rational agent

- **Performance measure :** An objective criterion for success of an agent's behavior
  - Awards one point for each clean square
    - at each time step, over 10000 time steps
- **Prior knowledge about the environment**
  - The geography of the environment
  - Only two squares
  - The *effect* of the actions
- **Actions that can perform**
  - Left, Right, Suck and NoOp
- **Percept sequences**
  - Where is the agent?
  - Whether the location contains dirt?
- Under this circumstance, the agent is rational.

Learning

- Does a rational agent depend on only current percept?
  - No, the past percept sequence should also be used
  - This is called **learning**
  - After experiencing an episode, the agent
    - should adjust its behaviors to perform better for the same job next time.
- If an agent just relies on the prior knowledge of its designer rather than its own percepts then the agent lacks ***autonomy***

A rational agent should be autonomous- it should learn what it can to compensate for partial or incorrect prior knowledge.

- E.g., a clock
  - No input (percepts)
  - Run only but its own algorithm (prior knowledge)
  - No learning, no experience, etc.

Software Agents

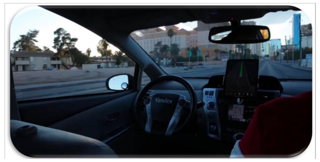
- Sometimes, the environment may not be the real world
  - E.g., flight simulator, video games, Internet
  - They are all artificial but very complex environments
  - Those agents working in these environments are called
    - Software agent (softbots)
    - Because all parts of the agent are software

Task environments

- **Task environments are the problems**
  - While the rational agents are the solutions
  - Must specify the task environment as fully as possible
- Specifying the task environment
  - **PEAS** description as fully as possible
    - Performance measure
    - Environment
    - Actuators
    - Sensors
- In designing an agent, the first step must always be to specify the task environment as fully as possible.

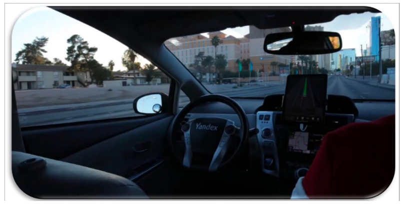
Task environments : Performance measure

- **Performance measure**
  - How can we **judge** the automated driver?
  - Which factors are considered?
    - getting to the correct destination
    - minimizing fuel consumption
    - minimizing the trip time and/or cost
    - minimizing the violations of traffic laws
    - maximizing the safety and comfort, etc.

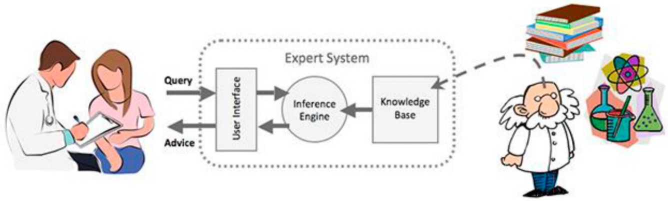


PEAS - Automated taxi driver

- Performance measure:
  - Safe, fast, legal, comfortable trip, maximize profits, impact on other road users
- Environment:
  - Roads, other traffic, pedestrians, customers, weather
- Actuators:
  - Steering wheel, accelerator, brake, signal, horn, display, speech
- Sensors:
  - Cameras, radar, sonar, speedometer, GPS, odometer, engine sensors, microphones, touchscreen



PEAS - Medical diagnosis system



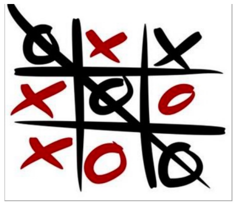
- Performance measure: Healthy patient, minimize costs, lawsuits
- Environment: Patient, hospital, staff
- Actuators: Screen display (questions, tests, diagnoses, treatments, referrals)
- Sensors: Keyboard (entry of symptoms, findings, patient's answers)

Environment types

- **Fully observable** (vs. partially observable):
  - If an agent's sensors give it access to the complete state of the environment at each point in time then the environment is effectively and fully observable
    - if the sensors detect all aspects
    - That are relevant to the choice of action
- Unobservable :
  - If the agent has no sensors at all
- **Partially observable:**
  - Due to noisy and inaccurate sensors, parts of the state are simply missing from the sensor data.

Example:

- A local dirt sensor of the cleaner cannot tell Whether other squares are clean or not

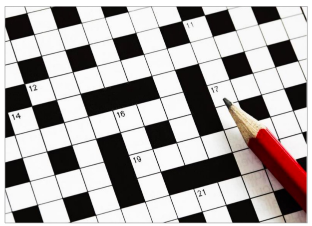


Environment types

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

-Cleaner and taxi driver are:

- Stochastic because of some unobservable aspects → noise or unknown



Environment types

- Episodic vs. sequential
  - The agent's experience is divided into atomic "episodes"
    - An episode = agent's single pair of perception & action
  - Eg: spot defective parts on an Assembly line
  - The quality of the agent's action does not depend on other episodes
    - Every episode is independent of each other
  - Episodic environment is simpler
    - The agent does not need to think ahead
- Sequential Environment
  - Current action may affect all future decisions
  - Ex. Taxi driving and chess.

Environment types

- Static vs. dynamic
  - The environment is unchanged while an agent is deliberating.
  - Agent need not keep looking at the world while deciding an action nor need it worry about the passage of time.
  - A dynamic environment is always changing over time
    - E.g., Taxi driving : the number of people in the street
- Semidynamic
  - environment is not changed over time
  - but the agent's performance score does
  - Eg: Chess when played with a clock is semi-dynamic

Rubik's Cube

vs.

Football playing robots

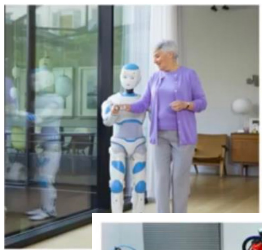
Environment types

- Discrete (vs. continuous):
- If there are a limited number of distinct states, clearly defined percepts and actions, the environment is discrete
  - E.g., Chess game
- Continuous: Taxi driving

VS.

Environment types

- Single agent (vs. multiagent):
- An agent operating by itself in an environment.
  - Playing a crossword puzzle – single agent
  - Chess playing – two agents
  - Competitive multiagent environment
    - Chess playing
  - Cooperative multiagent environment
    - Automated taxi driver
    - Avoiding collision



Environment types

- Known vs. unknown
- This distinction refers not to the environment itslef but to the agent's (or designer's) state of knowledge about the environment.
  - In known environment, the outcomes for all actions are given.
  - example: solitaire card games
- If the environment is unknown, the agent will have to learn how it works in order to make good decisions.
  - example: new video game.

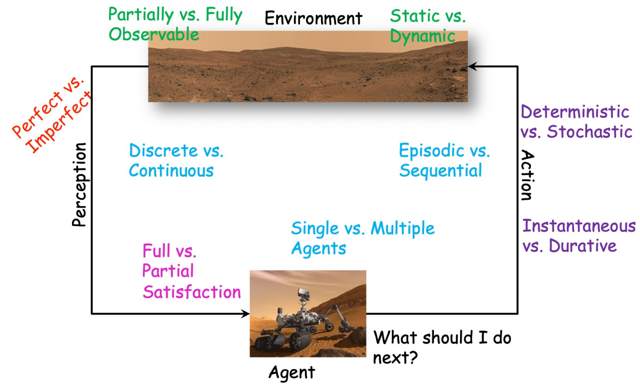
Examples of task environments

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

The environment type largely determines the agent design

- **The real world is : partially observable, stochastic, sequential, dynamic, nondeterministic, continuous, multi-agent . How do we handle it then?**

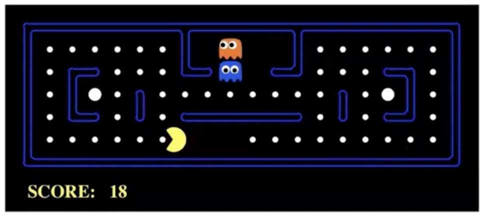
Properties of the Task Environment

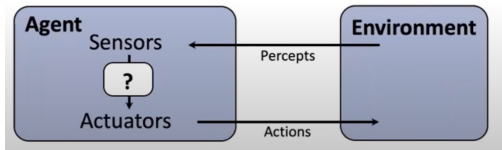


Structure of agents

- Agent = architecture + program
  - Architecture = some sort of computing device (sensors + actuators)
  - (Agent) Program = some function that implements the agent mapping = “?”
  - Agent Program = Job of AI

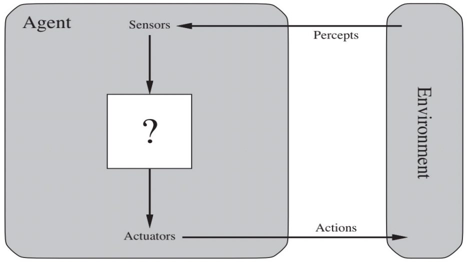
Pac-Man as an Agent





Agent functions and programs

- An agent is completely specified by the agent function mapping percept sequences to actions
- Input for Agent Program
  - Only the current percept
- Input for Agent Function
  - The entire percept sequence
  - The agent must remember all of them
- Implement the agent program as
  - A look up table (agent function)



Agent types /Agent architectures

- Five basic types in order of increasing generality
  - Table Driven agents
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents

Agent functions: Table-lookup agent

- Lookup Table – An action for every possible percept sequence.
- P = the set of possible percepts
- T= lifetime of the agent
  - The total number of percepts it receives
- Size of the look up table

$$\sum_{t=1}^{T} |P|^{t}$$

- Consider playing chess
  - P =10, T=150
  - Will require a table of at least 10^{150} entries

Table-lookup agent

- Skeleton design of an agent program

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

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

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
⋮	⋮

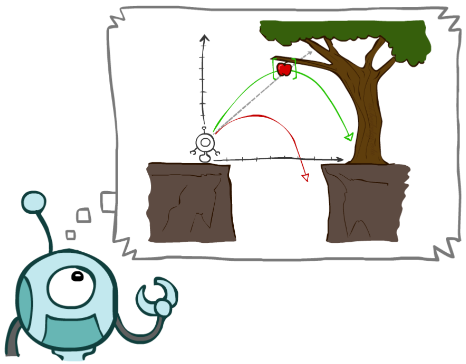
- \input{algorithms/table-agent-algorithm}
- Drawbacks: – Huge table
  - Take a long time to build the table
  - No autonomy
  - Even with learning, need a long time to learn the table entries

Agent programs

- Despite of huge size, look up table does what we want.
- The key challenge of AI
  - Find out how to write programs that, to the extent possible, produce rational behavior
    - From a small amount of code
    - Rather than a large amount of table entries
  - E.g., a five-line program of Newton's Method running on electronic calculators.
  - V.s. huge tables of square roots, sine, cosine, …

Types of agent programs

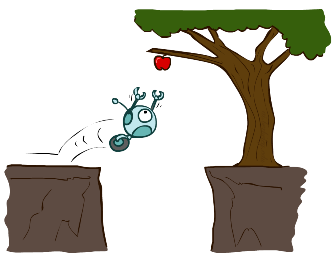
- Four type
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents



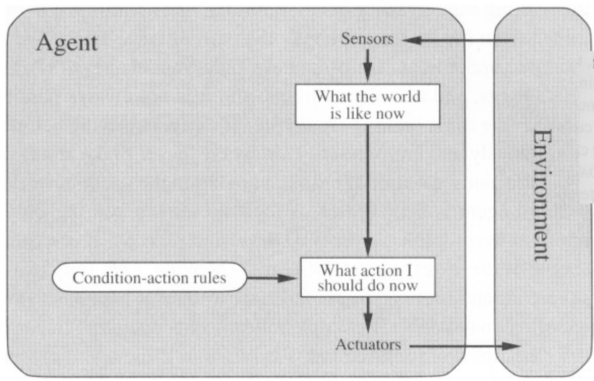
Simple reflex agents

- **Reflex agents:**
  - Choose action based on current percept (and maybe memory)
  - May have memory or a model of the world's current state
  - Do not consider the future consequences of their actions
  - Consider how the world IS

- *Can a reflex agent be rational?*
- It uses just condition-action rules
  - The rules are like the form "if … then …"
  - efficient but have narrow range of applicability
  - Because knowledge sometimes cannot be stated explicitly
  - Work only
    - if the environment is fully observable



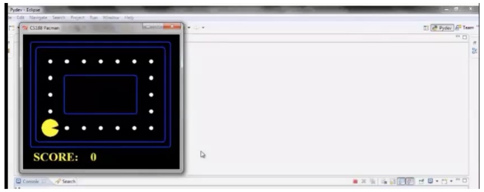
Simple reflex agents



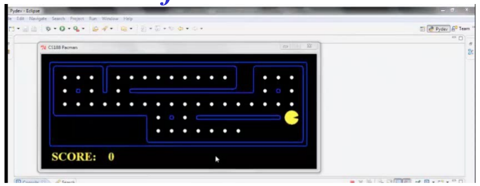

```
function SIMPLE-REFLEX-AGENT(percept) returns action
  static: rules, a set of condition-action rules

  state ← INTERPRET-INPUT(percept)
  rule ← RULE-MATCH(state, rules)
  action ← RULE-ACTION[rule]
  return action
```

Pac-Man : Video of Demo Reflex Optimal

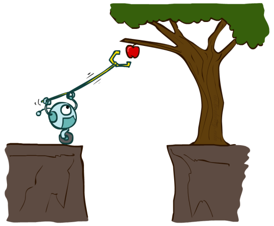


Not Good for other Cases

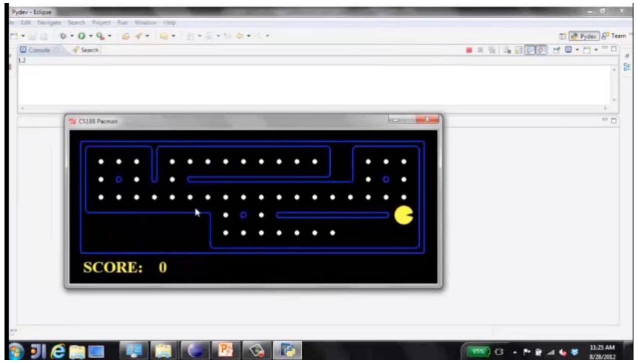


Planning Agents

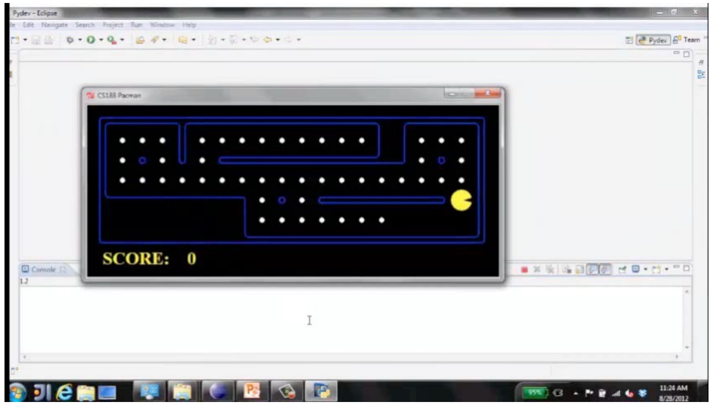
- Ask "what if"
- Decisions based on (hypothesized) consequences of actions
- Must have a model of how the world evolves in response to actions
- Must formulate a goal (test)
- Consider how the world WOULD BE
- Optimal vs. Complete planning
- Planning vs. replanning
  - Simulate many games, execute one. Doesn't do it in the world, does it in the model.
    - Complete – a solution; optimal – best



Demo Plan slow (Mastermind Pac-Man)



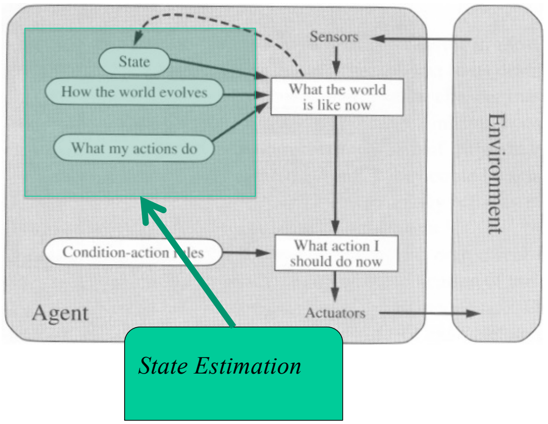
Video of Demo Plan fast ('replanning')



Model-based reflex agents

- Also known as State based Reflex Agents
- For the world that is partially observable
  - the agent has to keep track of an internal state
    - That depends on the percept history
    - Reflecting some of the unobserved aspects
    - E.g., driving a car and changing lane
- Based on state of the world and knowledge (memory), it triggers actions through the effectors
- Requiring two types of knowledge
  - How the world evolves independently of the agent
  - How the agent's actions affect the world

Model-based Reflex Agents



function REFLEX-AGENT-WITH-STATE(*percept*) **returns** *action*
static: *state*, a description of the current world state
rules, a set of condition-action rules

state ← UPDATE-STATE(*state*, *percept*)
rule ← RULE-MATCH(*state*, *rules*)
action ← RULE-ACTION[*rule*]
state ← UPDATE-STATE(*state*, *action*)
return *action*

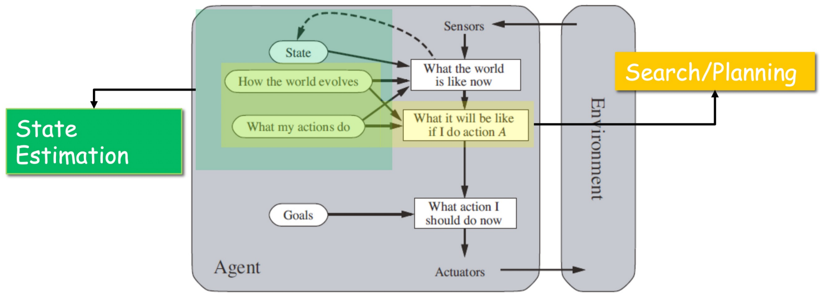
More Reasoning: a model of the world

The agent is with memory

Goal-based agents

- Current state of the environment is always not enough
- The goal is another issue to achieve
  - Judgment of rationality / correctness
- Actions chosen → goals, based on
  - the current state
  - the current percept
- Goal-based agents are less efficient but more flexible
  - Agent ←Different goal ←different tasks
- Search and planning
  - two other sub-fields in AI
  - to find out the action sequences to achieve its goal

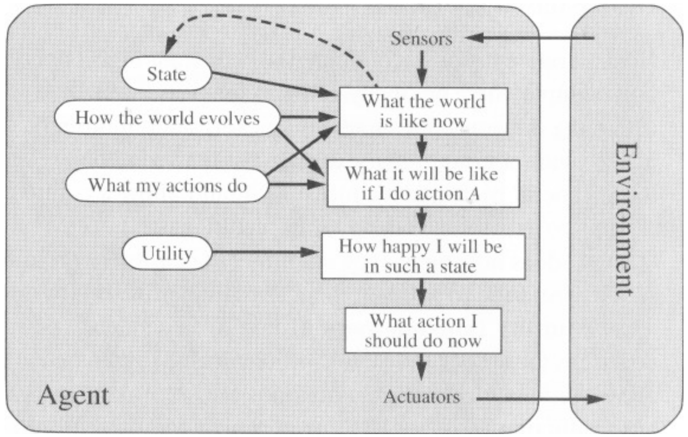
Goal based agents



Utility-based agents

- Goals alone are not enough
  - to generate **high-quality** behavior
  - E.g. meals in Canteen, good or not ?
- Many action sequences → the goals
  - some are better and some worse
  - If goal means success,
  - then **utility** means the degree of success (how successful it is)
- It is said state A has higher utility
  - If state A is more preferred than others
- Utility is therefore a function
  - that maps a state onto a real number
  - the degree of success

Utility-based agents



Utility-based agents

- Utility has several advantages:
  - When there are conflicting goals,
    - Only some of the goals but not all can be achieved
    - utility describes the appropriate trade-off
  - When there are several goals
    - None of them are achieved certainly
    - utility provides a way for the decision-making

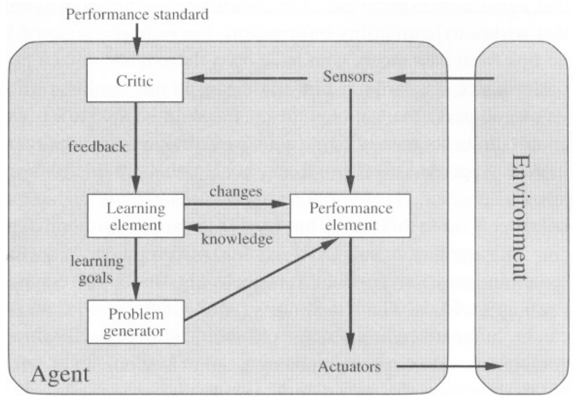
Learning Agents

- After an agent is programmed, can it work immediately?
  - No, it still need teaching
- In AI,
  - Once an agent is done
  - We teach it by giving it a set of examples
  - Test it by using another set of examples
- We then say the agent learns
  - A learning agent

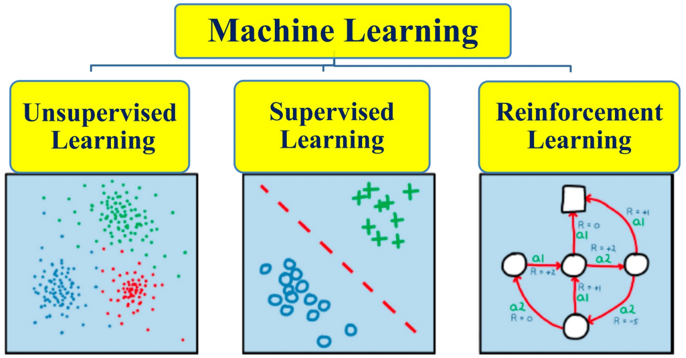
Learning Agents

- Four conceptual components
  - Learning element
    - Making improvement
  - Performance element
    - Selecting external actions
  - Critic
    - Tells the Learning element how well the agent is doing with respect to fixed performance standard.
    - (Feedback from user or examples, good or not?)
  - Problem generator
    - Suggest actions that will lead to new and informative experiences.

Learning Agents

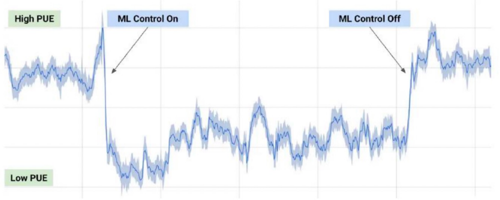


Three broad categories of ML



Google Data Centre Cooling

- DeepMind AI Reduces Google Data Centre Cooling Bill by 40%
- Optimal operation of pumps, chillers and cooling towers
- Compared to five years ago, Google get around 3.5 times the computing power out of the same amount of energy



Reference: Emilie Kaufmann et al. Adaptive Reward-Free Exploration
https://arxiv.org/pdf/2006.06294.pdf

Ideal Rational Agent

- For each possible percept sequence, does whatever action is expected to maximize its performance measure on the basis of evidence perceived so far and built-in knowledge

Agent design

The environment type largely determines the agent design

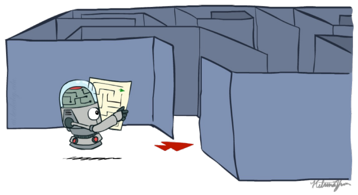
- Partially observable => agent requires memory (internal state)
- Stochastic => agent may have to prepare for contingencies
- Multi-agent => agent may need to behave randomly
- Static => agent has time to compute a rational decision
- Continuous time => continuously operating controller
- Unknown physics => need for exploration
- Unknown perf. measure => observe/interact with human principal

What is Artifical Intelligence

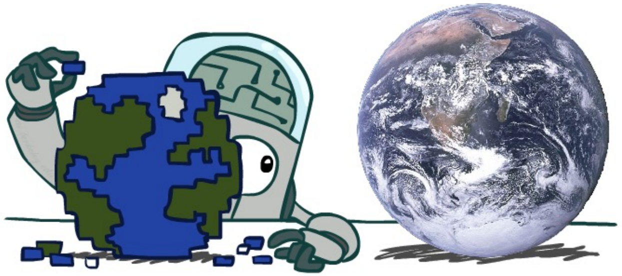
- Algorithmic view
  - A large number if problem are NP hard
  - AI develops a set of tools, heuristics,
    - to slove such problems in practice
    - for naturally occurring instances
  - Search
  - Game Playing
  - Planning
  - ....

Complex Problems

Our aim is to solve all type of problems



Search Problems Are Models



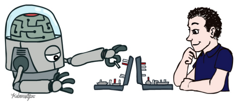
Conclusion

Next

- Module 2: Automated Problem Solving
  - PART 2.1: Intelligent Agent & Environment
  - PART 2.2: Complex Problems and AI
  - PART 2.3: Problem Solving Methods
-

References

- Slides adapted from CS188 Instructor: Anca Dragan, University of California, Berkeley
- Slides adapted from CS60045 ARTIFICIAL INTELLIGENCE



(some slides adapted from http://aima.cs.berkeley.edu/)