CMSC 372
Artificial Intelligence
Spring 2015

Landmarks in AI History

- 1943: McCulloch & Pitts
  Boolean Circuit model of brain
- 1950: Alan Turing
  Computing Machinery & Intelligence
- 1956: Birth of Artificial Intelligence
  Dartmouth Conference
- 1952-69: Look Ma, no hands!
  GPS, Geometry Probs, Solver, Chessers, USP
Landmarks in AI History

1940
- McCulloch & Pitts: Boolean Circuit model of brain

1943
- Alan Turing: Computing Machinery & Intelligence

1943-49
- Look Ma, no hands! GPS, Geometry Prob. Solver, Checkers, LISP

1949
- Robinson: Algorithm for logical reasoning

1950
- Minsky & Papert: Perceptrons & Neural Network Agenda

1956
- Birth of Artificial Intelligence: Dartmouth Conference

1956
- ALPAC Report: Machine Translation Killed

1956
- Block et al: Perceptron Convergence Theorem

1957
- Newell & Simon: Physical Symbol System Hypothesis

1958
- Resurgence of probability

1959
- Minsky & Papert: Perceptrons kill Neural Network Agenda

1959-79
- Knowledge-based systems: DENDRAL, MYCIN, SHIRL, PLANER, C, frames

1962
- Block et al: Perceptron Convergence Theorem

1965
- Robinson: Algorithm for logical reasoning

1965
- Robinson: Algorithm for logical reasoning

1966
- ALPAC Report: Machine Translation Killed

1969
- Minsky & Papert: Perceptrons kill Neural Network Agenda

1969-79
- Knowledge-based systems: DENDRAL, MYCIN, SHIRL, PLANER, C, frames

1976
- Newell & Simon: Physical Symbol System Hypothesis

1980
- AI becomes an industry

1980-95
- Rebirth of Neural networks: PDP, Connectionist models, backprop

1985-95
- Resurgence of probability

1988
- Resurgence of probability

1988
- Newell & Simon: Physical Symbol System Hypothesis

1990
- AI Winter

1990-95
- Expert Systems go bust

1995
- Agents everywhere!

2001
- AI Spring

2001
- AI Spring

2006
- AI yields advances

2006
- AI yields advances

2011
- Big Data AI

2011
- Big Data AI

2011
- Big Data AI

2011
- Big Data AI

2011
- Big Data AI
 Agenda

• What is AI? History, Foundations, Examples: Overview
• Intelligent Agents
• Problem Solving Using Classical Search Techniques
• Beyond Classical Search
• Adversarial Search & Game Playing
• Constraint Satisfaction Problems
• Knowledge Representation & Reasoning (KRR)
• First Order Logic & Inference
• Classical Planning
• Planning & Acting in the Real World
• Other topics depending upon time...

AI: State of the Art

Chapter 1, Exercise 1.14: Which of the following can be solved by computers?

• Play a decent game of table tennis (Ping Pong)
• Driving autonomously in Bryn Mawr, PA
• Driving in Cairo
• Buy a week's worth of groceries at the market
• Buy a week's worth of groceries on the web
• Play a game of bridge at the competitive level
• Discovering and proving new mathematical theorems
• Write an intentionally funny story
• Giving competent legal advice in a specialized area of law
• Translate spoken English into spoken Swedish in real time
• Perform a complex surgical operation
AI: State of the Art

Chapter 1, Exercise 1.14: Which of the following can be solved by computers?

• Play a decent game of table tennis (Ping Pong)
• Driving autonomously in Bryn Mawr, PA
• Driving in Cairo
• Buy a week's worth of groceries at the market
• Buy a week's worth of groceries on the web
• Play a game of bridge at the competitive level
• Discovering and proving new mathematical theorems
• Write an intentionally funny story
• Giving competent legal advice in a specialized area of law
• Translate spoken English into spoken Swedish in real time
• Perform a complex surgical operation

Intelligent Agents

• Agents and environments
• Rationality
• PEAS
  (Performance measure, Environment, Actuators, Sensors)
• Environment types
• Agent types
What is AI?

Rational Behavior

- Do the right thing.
- That which is expected to maximize goal achievement, given available information.
- Doesn’t necessarily involve ‘thinking’. E.g. blinking reflex.
- Any thinking there is, should be in service of rational action.
- Design Rational Agents.

\[ f: P^* \rightarrow A \]

Problem: Computational limitations make perfect rationality unachievable. So, design best program for given computational resources.

Agents

- **Agent**: 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

- **Software agent**: receives keystrokes, file contents, network packets as sensory inputs and acts by displaying, writing files, sending network packets, etc.
Agents

- **Agent**: 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.

- **Software agent**: receives keystrokes, file contents, network packets as sensory inputs and acts by displaying, writing files, sending network packets, etc.

Agents

- **Perception**: sensors
- **Actions**: actuators
- **Environment**: the world the agent is in
Agent = Architecture + Program

- The agent function maps from percept histories to actions:
  \[ f: \mathcal{P}^* \rightarrow \mathcal{A} \]
- The agent program runs on the physical architecture to produce \( f \)

Example: Vacuum-cleaner world

- Percepts: location and contents, e.g., [A,Dirty]
- Actions: Left, Right, Suck
### A vacuum-cleaner agent

<table>
<thead>
<tr>
<th>Percept Sequence</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A, Clean]</td>
<td>Right</td>
</tr>
<tr>
<td>[A, Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>[B, Clean]</td>
<td>Left</td>
</tr>
<tr>
<td>[B, Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>[A, Clean], [A, Clean]</td>
<td>Right</td>
</tr>
<tr>
<td>[A, Clean], [A, Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**Agent Program**

```plaintext
function TableDrivenVacuumAgent(percept) returns action
append percept to end of percepts
action ← LookUp(percepts, table)
return action
```

percepts

...
A vacuum-cleaner agent

<table>
<thead>
<tr>
<th>Percept Sequence</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A, Clean]</td>
<td>Right</td>
</tr>
<tr>
<td>[A, Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>[B, Clean]</td>
<td>Left</td>
</tr>
<tr>
<td>[B, Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>[A, Clean], [A, Clean]</td>
<td>Right</td>
</tr>
<tr>
<td>[A, Clean], [A, Dirty]</td>
<td>Suck</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

function ReflexVacuumAgent([location, status]) returns action
if status = Dirty then return Suck
else if location = A then return Right
else if location = B then return Left

Designing Agents

• What is the right function?

• Can it be implemented in a small agent program?

• Is this a good agent? Bad? Stupid?...analysis!
Analysis: Performance Measure

- An agent should strive to "do the right thing", based on what it can perceive and the actions it can perform. The right action is the one that will cause the agent to be most successful.

- **Performance measure**: An objective criterion for success of an agent's behavior.

- E.g., performance measure of a vacuum-cleaner agent could be amount of dirt cleaned up, amount of time taken, amount of electricity consumed, amount of noise generated, etc.

Rational Agents

- **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.
Rational Agents

• Rationality is distinct from *omniscience* (all-knowing with infinite knowledge)

• 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)

PEAS

• **PEAS**: Performance measure, Environment, Actuators, Sensors

• Must first specify the setting for intelligent agent design

• Consider, e.g., the task of designing an automated taxi driver
  
  – Performance measure
  – Environment
  – Actuators
  – Sensors
**PEAS**

- **PEAS**: Performance measure, Environment, Actuators, Sensors

- Must first specify the setting for intelligent agent design

- Consider, e.g., the task of designing an automated taxi driver
  - **Performance measure**: Safe, fast, legal, comfortable trip, profits
  - **Environment**: Roads, other traffic, pedestrians, customers
  - **Actuators**: Steering wheel, accelerator, brake, signal, horn
  - **Sensors**: Cameras, sonar, speedometer, GPS, odometer, engine sensors, keyboard

**PEAS**

- **Agent**: 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)
PEAS

- **Agent**: Part-picking robot
- **Performance measure**: Percentage of parts in correct bins
- **Environment**: Conveyor belt with parts, bins
- **Actuators**: Jointed arm and hand
- **Sensors**: Camera, joint angle sensors

Environment Types

- **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)
- **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.
Environment Types

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

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

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

<table>
<thead>
<tr>
<th>Observable?</th>
<th>Chess w/o clock</th>
<th>Solitaire</th>
<th>Internet Shopping</th>
<th>Taxi</th>
<th>Real World</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Episodic?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Static?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discrete?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single agent?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Environment Types

## Environment Types

<table>
<thead>
<tr>
<th></th>
<th>Chess w/o clock</th>
<th>Solitaire</th>
<th>Internet Shopping</th>
<th>Taxi</th>
<th>Real World</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observable?</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deterministic?</td>
<td>Strategic</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Episodic?</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Static?</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discrete?</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single agent?</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Environment Types

<table>
<thead>
<tr>
<th></th>
<th>Chess w/o clock</th>
<th>Solitaire</th>
<th>Internet Shopping</th>
<th>Taxi</th>
<th>Real World</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observable?</td>
<td>Yes</td>
<td>Partly</td>
<td>No</td>
<td>Partly</td>
<td>Partly</td>
</tr>
<tr>
<td>Deterministic?</td>
<td>Strategic</td>
<td>Yes</td>
<td>Partly</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Episodic?</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Static?</td>
<td>Yes</td>
<td>Yes</td>
<td>Semi</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Discrete?</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Single agent?</td>
<td>No</td>
<td>Yes</td>
<td>Yes, but...</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Environment Types

<table>
<thead>
<tr>
<th></th>
<th>Chess w/o clock</th>
<th>Solitaire</th>
<th>Internet Shopping</th>
<th>Taxi</th>
<th>Real World</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observable?</td>
<td>Yes</td>
<td>Partly</td>
<td>No</td>
<td>Partly</td>
<td>Partly</td>
</tr>
<tr>
<td>Deterministic?</td>
<td>Strategic</td>
<td>Yes</td>
<td>Partly</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Episodic?</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Static?</td>
<td>Yes</td>
<td>Yes</td>
<td>Semi</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Discrete?</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Single agent?</td>
<td>No</td>
<td>Yes</td>
<td>Yes, but...</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Environment type determines agent design.

Agent functions and programs

- An agent is completely specified by the agent function mapping percept sequences to actions
- One agent function (or a small equivalence class) is rational
- **Aim:** find a way to implement the rational agent function concisely
Table-Lookup Agent

function TableDrivenVacuumAgent(percept) returns action
append percept to end of percepts
action ← LookUp(percepts, table)
return action

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

Reflex Vacuum Agent

function ReflexVacuumAgent([location, status]) returns action
if status = Dirty then return Suck
else if location = A then return Right
else if location = B then return Left
Agent Types

In order of increasing generality:

- Simple reflex agents
- Model-based reflex agents
- Goal-based agents
- Utility-based agents
Agent Types

In order of increasing generality:

• Simple reflex agents

• Model-based reflex agents

• Goal-based agents

• Utility-based agents

Simple Reflex Agents
Simple Reflex Agents

function SimpleReflexAgent(percept) returns Action
persistent: rules, a set of condition-action rules

state ← InterpretInput(percept)
rule ← RuleMatch(state, rules)
action ← rule.Action
return action

Model-Based Reflex Agents

![Diagram of Model-Based Reflex Agents]

Agent

- State
- Condition-Action Rules

Sensors
- How the world evolves
- What the world is like now
- What my actions do

Environment

- What action I should do now

Actuators
Model-Based Reflex Agents

```plaintext
function ModelBasedReflexAgent(percept) returns Action
persistent:
    state, the agent’s current conception of the world state
    model, a description of how the next state depends
        on current state and action
    rules, a set of condition-action rules
    action, the most recent action, initially none

state ← UpdateState(state, action, percept, model)
rule ← RuleMatch(state, rules)
action ← rule.Action
return action
```

Goal-Based Agents
Utility-Based Agents

State

- How the world evolves
- What my actions do
- What the world is like now
- What it will be like if I do A
- How happy I will be in such a state
- What action I should do now

Utility

Actuators

Environment

Learning Agents

Critic

- Feedback
- Changes
- Knowledge
- Learning goals

Problem generator

Learning element

Performance element

Actuators

Environment
Summary

- Agents interact with environments through sensors and actuators
- Agent function describes what the agent does in all circumstances
- Performance measure evaluates the environment sequence
- A perfectly rational agent maximizes expected performance
- Agent programs implement (some) agent functions
- PEAS descriptions define task environments
- Environments are categorized along several dimensions: observable? deterministic? episodic? static? discrete? single-agent?
- Several basic agent types exist: reflex, reflex with state (model), goal-based, utility-based

Acknowledgements

- This presentation is being made available by Deepak Kumar for any and all educational purposes. Please feel free to use, modify, or distribute. Powerpoint file(s) are available upon request by writing to dkumar@cs.brynmawr.edu
- Prepared in January 2015.