Turing Test

Alan Turing 1912-1954

- Important theory of computation
  - Mathematical model of computer (Turing machine)
    - Any "Turing-complete" computer can simulate time and memory), so they're all equivalent
  - Some problems are unsolvable
    - Halting problem
- During WWII, developed machine to break German codes
  - Knighted (OBE), 1946
- Convicted of "gross indecency" for homosexuality, 1952
  - Chose probation and hormone injections over prison
  - Denied entry into US as a result
- Died of cyanide poisoning, 1954
  - Ruled suicide, but there is some debate
- Pardoned by the Queen in 2014

Imitation Game

- Loebner prize

Contrary Views

- "I believe that at the end of the century the use of words and general educated option will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted."
- Theological Objection
  - Turing is unimpressed
- Heads in the Sand Objection
  - Singularity
- Mathematical Objection
  - Gödel's theorem
    - Any sufficiently powerful mathematical system contains true statements that cannot be proven
  - Uncomputability
  - No basis for assertion that humans are immune to this
    - Collatz conjecture
- Argument from Consciousness
  - How do we know other people think?
  - Thinking vs "artificially signaling"
  - Sealer's Chinese Room

- Arguments from Various Disabilities
  - No machine has done X, so no machine can
  - Strange comment about friendliness "between white man and white man, or between black man and black man"
  - Machines cannot make mistakes
    - They could deliberately do so
  - Computers can "think" about their own programming
- Lady Lovelace's Objection
  - Charles Babbage 1791-1871
    - Difference engine, designed 1822
      - Never build in his lifetime
      - One was finished in 1991
    - Analytical engine
      - Programmed using punched cards
      - Turing-complete Also never built
  - Ada Lovelace 1815-1852
    - Worked with Babbage
    - Studies mathematics to keep her from succumbing to poetry
      - Her father was Lord Byron
    - In 1840, Babbage gave a talk on the analytical engine in Italy
      - An Italian engineer wrote up the lecture in French
      - Ada was tasked with translating it into English
      - She added extensive notes
      - One contained the first computer program
    - A computer can only do what we tell it to do, not create anything new
      - Is anything really new?
      - Computers surprised Turing all the time
      - "The assumption that as soon as a fact is presented to a mind all useful consequences of that fact spring into the mind simultaneously with it."
- Argument from Continuity in the Nervous System
  - Meh, you can simulate floating point numbers
- Argument from Informality of Behavior
  - Rules -> machine, no rules for people, therefore people are not machines
    - Invalid logic
  - Laws of behavior vs rules of conduct
    - How do we know there are no laws of behavior?
  - Having laws of behavior does not imply predictability
    - Pseudo random number generator
- Argument form Extrasensory Perception
  - Turing finds this compelling!
  - Solution: telepathy-proof room

Learning Machines

- Elitism: most minds are sub critical
- Skin-of-an-onion analogy fro discussing minds
- "the problem is mainly one of programming. Advances in engineering will have to be made too."
- Develop a child machine and teach it
  - Analogy with natural selection, foreshadowing genetic algorithms
  - Rewards and punishments Later, communication through symbolic language
  - Building in a system of logical interface
  - Can change higher-level, ephemeral rules
  - The "teacher will often be very largely ignorant of quite what is going on inside"
- What tasks to tackle?
  - Chess?
  - English?

What is Artificial Intelligence?

What are we trying to accomplish?

- Thinking vs acting
- Humanly vs rationally

Humanity	Think	Act
Humanly	Brain modeling	pass Turing test
Rationally	not too useful in messy real world	YES!

Rational Agents

Agent acts in an environment

- Receives a sequence of precepts through sensors
- Affects environment by taking actions through actuators
- Performance measure indicates how well the agent is doing

PEAS (Performance, Environment, Actuators, Sensors)

Properties of environments

- Fully observable vs partially observable vs unobservable
- Single-agent vs multi-agent
- Deterministic vs stochastic
- Static vs dynamic vs semi dynamic
- Discrete vs continuous
- Known vs unknown
  - This is about knowing the "rules" in advance

Rationality

- Does not demand *omniscience* – agent isn't responsible for information it would have no way of knowing
  - This doesn't excuse failure to explore
- Rationality is defined poorly in terms of behavior
  - Learning is neither necessary nor sufficient

Types of agents

- Agent function maps precepts sequences to actions
  - Two-location vacuum world example
  - Implemented by agent program
- **Reflex**: Always responds same way to a given percent
  - Randomization may be helpful
    - Avoid infinite loop in two-location vacuum world
    - Rock-Paper-Scissors
- **State-Based**: Has memory, which can be affected by past precepts
  - Could range form counting to building a map/model of the world
- **Goal-Based**: Has internal sense of which states are desirable
  - Chooses actions to reach these states
  - Reasons about how actions will affect state
- **Utility-Based**: Has internal sense of performance measure
  - Chooses actions to maximize utility
- **Learning**: Adjusts is program based on result of past actions
  - Perhaps it receives rewards or punishments
  - May need to balance exploration and exploitation

Uniform Search

Problem-solving agents

- For problems that are observable, single-agent, deterministic, static, discrete, and known.
- Agent observes state, formulates plan, and then carries out the plan.
  - No feedback from environment while carrying out plan
- Problem definition
  - Initial state
  - Actions function (maps state to available actions)
  - Transition model (maps state, action pair to successor state)
    - The states therefor form a graph
  - Goal test
  - Path cost function
- Solution is a path (series of actions) from initial state to goal state.

Searching for solutions

- Build a search tree
  - Notes correspond to states
  - Expanding a node means creating its children
  - All leaf nodes form the frontier
  - General tree search algorithm
    - Variability comes in which node to expand next
  - Problem: cycles and redundant paths
    - Cycles can lead to an infinite loop
    - Redundant paths can lead to a lot of extra work
    - Solution: maintain an explored set
    - The frontier separates the explored and unexplored regions

Uninformed search strategies

- Breadth-first search
  - Expand oldest node first
- Uniform cost search
  - Similar, but expands node with lowest path cost
- Depth-first search
  - Expand newest node first
- Depth-limited search
  - Just like DPF, but you limit the depth
- Iterative deepening depth-first search

  - Repeatedly do depth-limited search, but with a larger depth each time
  - This is the preferred uninformed search method

Heuristic Search

Best-first search

- Choose node to expand based on some function $f(n)$
- Uniform-cost search uses $f(n) = g(n)$
- Greedy best-first search: use $f(n) = h(n)$, estimated remaining cost
  - Greedy because it does what seems best right now
  - Tree search is incomplete
  - Graph search is complete for finite spaces
  - Worst case time and space is $O(b^m)$, where m is maximum depth of search space.
- A* search uses $f(n) = g(n) + h(n)$
  - This is the estimated cost of the cheapest path through n
  - Is it optimal? Depends on the heuristic
    - An admissible heuristic never overestimates the cost
      - It is optimistic
    - Consistent heuristic satisfies $h(n) >= c(n, a, n') + h(n')$, n' being a successor of n
      - This is a version of the triangle inequality
      - All consistent heuristics are admissible
      - "One has to work quite are to concoct heuristics that are admissible but not consistent"
      - Straight-line distance is both admissible and consistent
    - A* tree search is optimal if $h(n)$ is admissible
    - A* graph search is optimal if $h(n)$ is consistent
      - If $h(n)$ is consistent, then the values of $f(n)$ along any path are non decreasing
      - Whenever A* selects a node n for expansion, the optimal path to that node has been found.
      - Nodes are expanded in increasing order of f(n)
  - Alternate interpretation: contours
    - Search works out from start node and never expands anything in a contour beyond the optional goal
    - "With more accurate heuristics, the bands will search toward the goal state and become more narrowly focused around the optimal path"
    - A* is optimally efficient – no algorithm expands fewer nodes
  - Time complexity depends on the quality of the heuristic
  - Space is still an issue for large problems

Designing heuristics

- Relaxing problems
  - Straight-line distance removes the constraint that you have to travel along roads
- What if we have several heuristics, none of which dominates any of the others?
  - Take the maximum, which dominates all of its components

Minimax Search

Variations

- Opening book
- Endgame book
- Depth-limited search
- Transposition table
- Alpha-beta pruning
- Heuristic move suggestion

Monte Carlo Tree Search

k-armed bandit problem

- Definition: Each pull either pays or doesn't
- Question: how to allocate pulls to maximize return?
- Exploration vs exploitation
- Solution: upper confidence bounds
- One version of formula

$$\frac{wins}{pulls} + \frac{explore}{pulls}$$

  Where *explore* is a constant
  - To avoid division by zero, we initialize each arm with an imaginary 1 win and 2 pulls
  - When there aren't many pulls, the second term can keep under explored arms in the running
  - As pulls -> infinity, the second term vanishes

Monte Carlo techniques in general

- Use in a wide variety of problems

- Idea: take many random samples

Monte Carlo tree search

- No need for static evaluation function
- Actually builds a search tree
- Note that minimax search does not
- For each turn in real game:
- Create the root node
- Thousands of times:
  - Play a simulated game from the current game state
  - Update search tree
- Choose best move from root
- A variation on best-first search, so some branches will be explored more deeply than others.
- Treats each node as a k-armed bandit problem
- Each node knows, for each of its children:
- Number of "pulls"
- Number of wins in simulated games
- How to play each simulated game?
- Start at the root and use the upper confidence bound formula to choose a child
- If you fall off the bottom of the tree, add a new leaf
- Play random moves until the end of the game
- Update winds and pulls for each node visited in this simulation
- Consequences:
- Approximation becomes better as a number of simulations increases
- If a move initially appears good but there is a strong refutation, its win rate will increase and then drop off as simulations are focused on the proper refutation
- Good enough time, would thoroughly explore the tree and spend all of its time on the best path