Algorithms for Modern Artificial Intelligence (NeoScholar Spring Program)

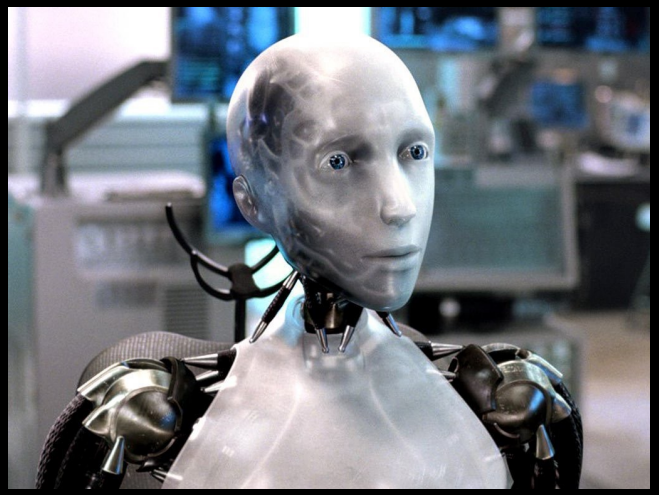
Lec1: An Introduction to AI

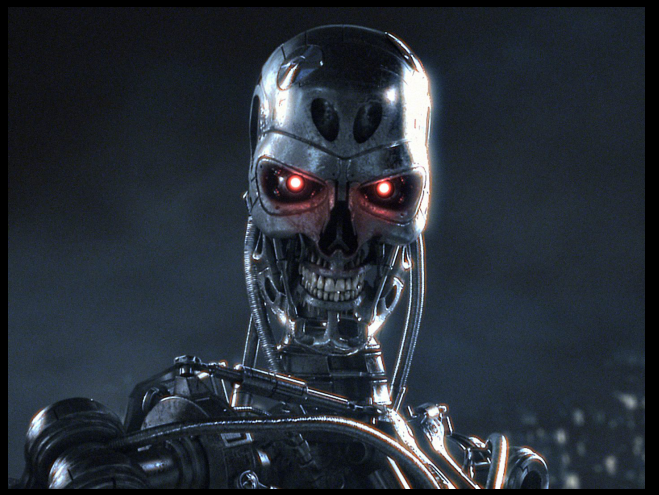
Haifeng Xu

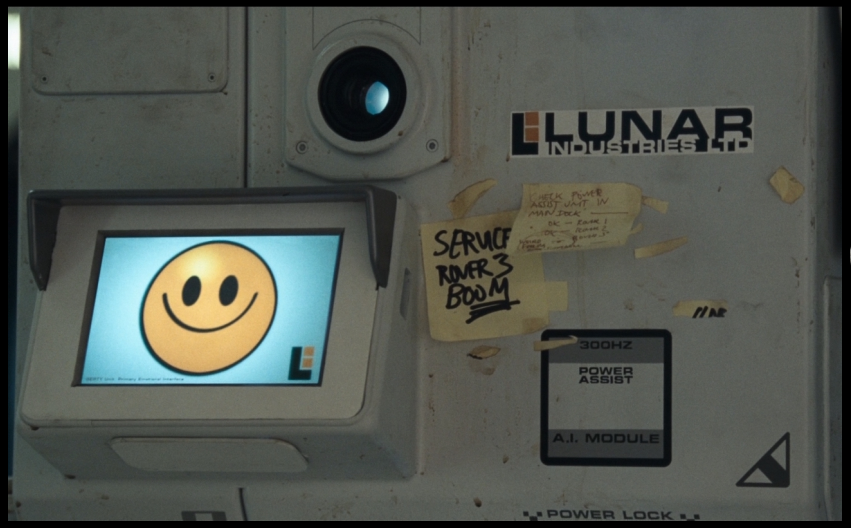
Professor of Computer Science and Data Science

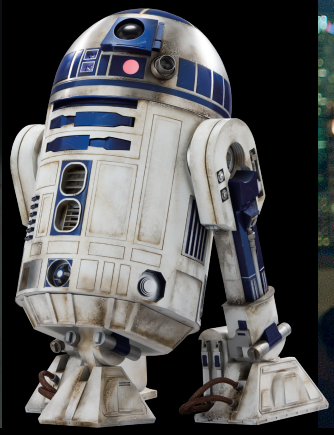
University of Chicago

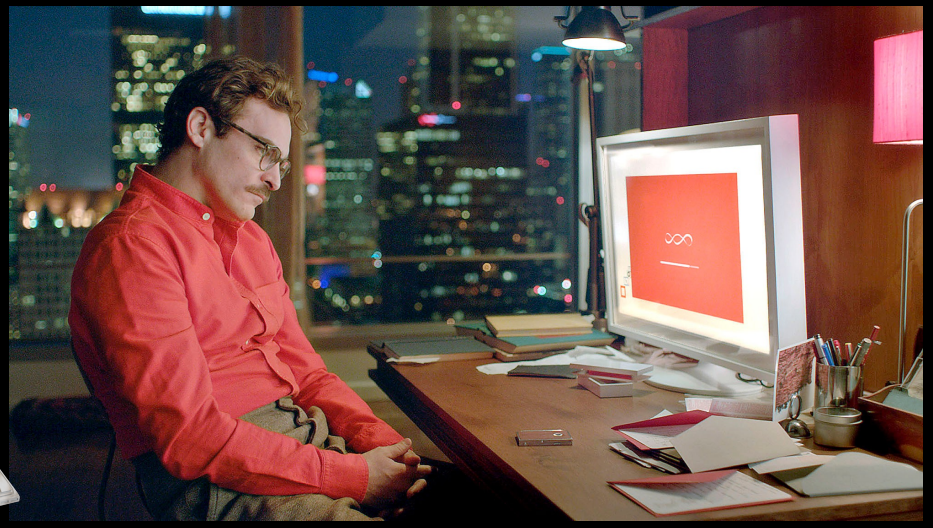
Artificial Intelligence











Artificial Intelligence

Def 0. The design of computer algorithms that **reproduce human behavior**

The Turing Test (1950)

Can a chatbot fool an informed observer?

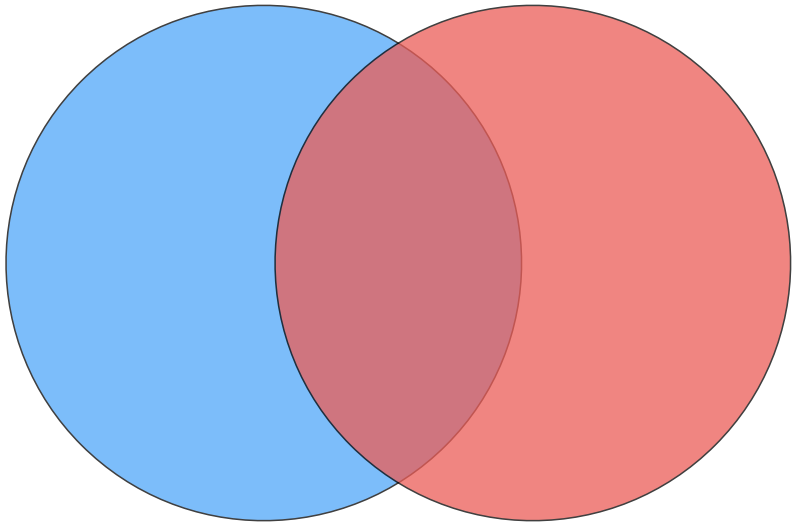
Alan Turing (1912-1954)

Artificial Intelligence

~~Def 0. The design of computer algorithms that reproduce human behavior~~

The Loebner Prize

A yearly Turing test competition for chatbots…



Artificial Intelligence

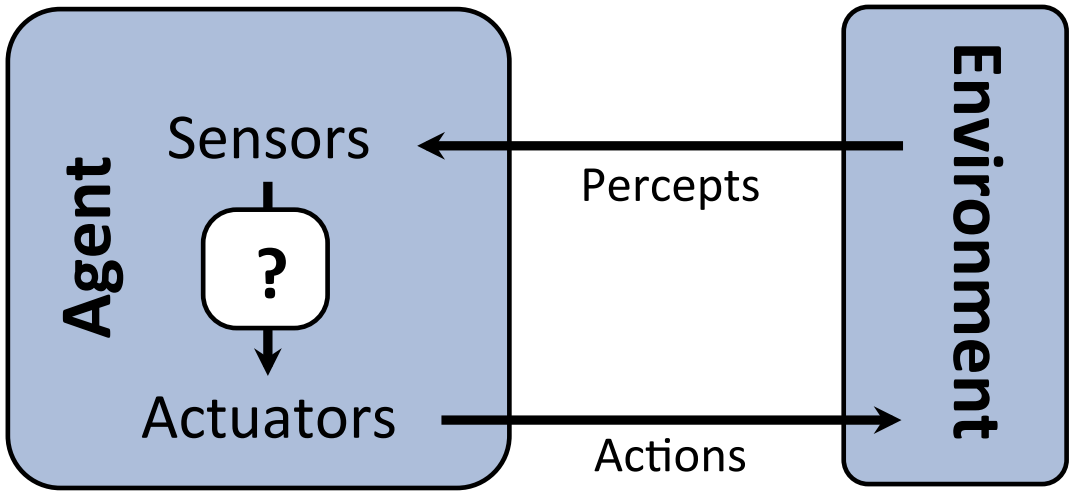
Def. The design of computer algorithms that **behave rationally with respect to a desired outcome**

Rationality

- optimally achieving objectives (minimizing cost, maximizing utility)
- is about the actions taken, not just the algorithmic steps that lead to decisions

The Pursue of Rational Agents

- An **agent** is a system that perceives the environment with **sensors** and changes the environment through **actions**.
  - Sense, ?, act, loop…
- A **rational agent** selects actions that maximize **utility**.
- Details of **environment**, **percepts**, and **action space** dictate different techniques for design of rational agents.
  - Deterministic vs. stochastic
  - Fully vs. partially observable
  - Discrete vs. continuous



The Thinking Machine (1961)

http://video.mit.edu/watch/the-thinking-machine-1961-mit-centennial-film-6712/

Nutshell History of AI

- 1940-1950: Early days
  - 1943: McCulloch & Pitts: Boolean artificial neurons
  - 1950: Turing's paper; Minsky & Edmonds Neural Net computer
- 1950—70: AI is born, excitement ensues!
  - 1956: Dartmouth meeting: the name “Artificial Intelligence” is adopted
  - 1950s: Samuel’s checkers program, Newell & Simon's Logic Theorist, Gelernter's Geometry Engine
- 1970—90: Knowledge-based approaches
  - 1969—79: Early development of knowledge-based systems
  - 1980—88: Expert systems industry booms
  - 1988—93: Expert systems industry busts: “AI Winter”
- 1990—: Statistical approaches; scientific method
  - Resurgence of probability, focus on uncertainty
  - General increase in technical depth; scientific method
- 2010—: Big data, fast computing and large model

Examples of AI technology

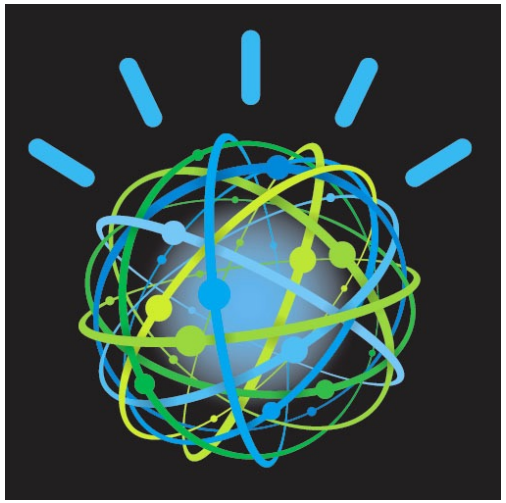
Table tennis

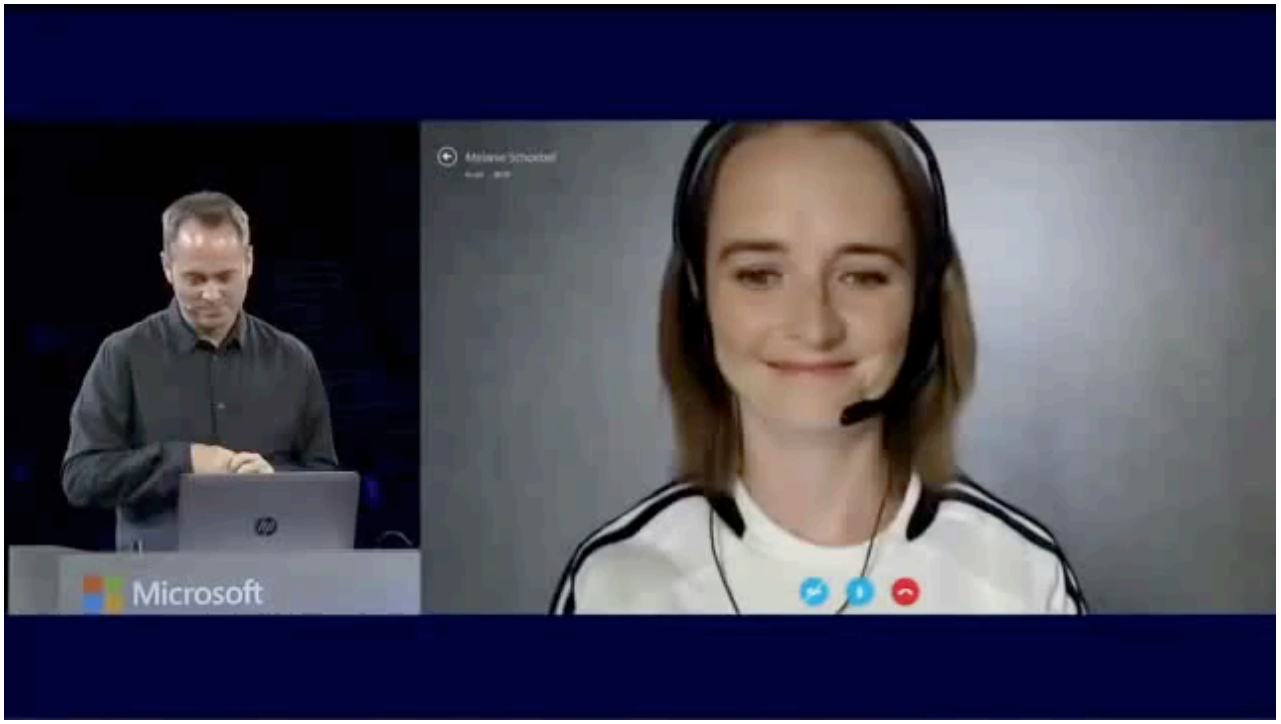
Muelling et al. 2010

[weeks 5/6]

Real-time Translation

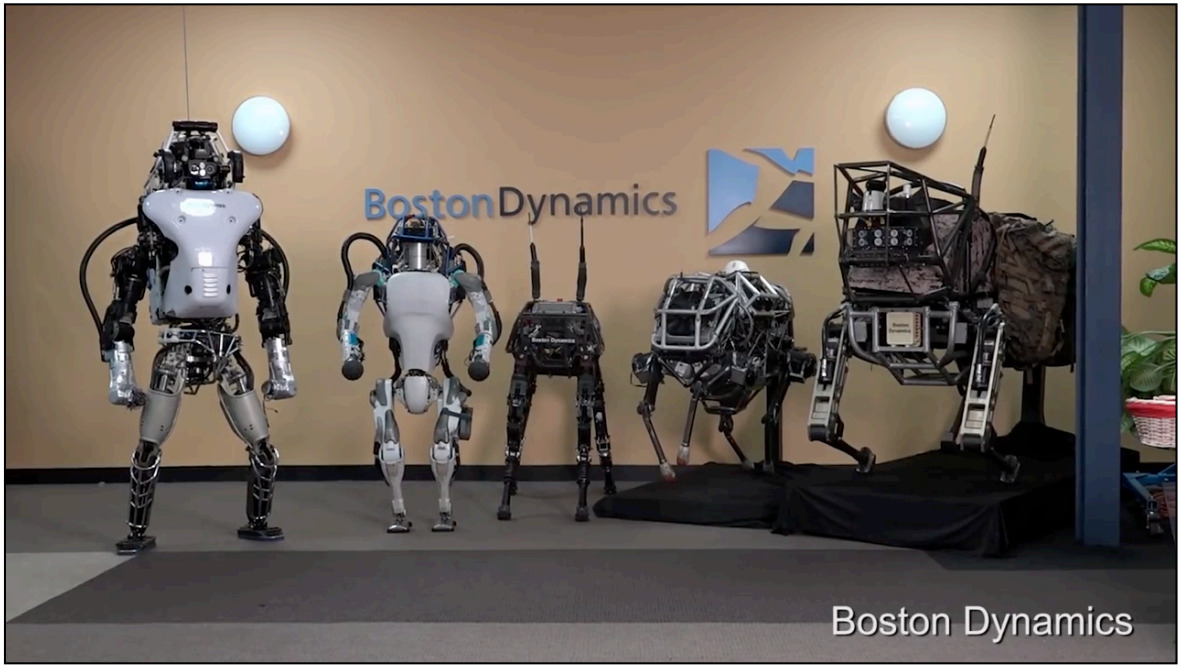
- Lots of data!
- Watson
  - Being applied to medical diagnosis with initial success
- Skype Translator





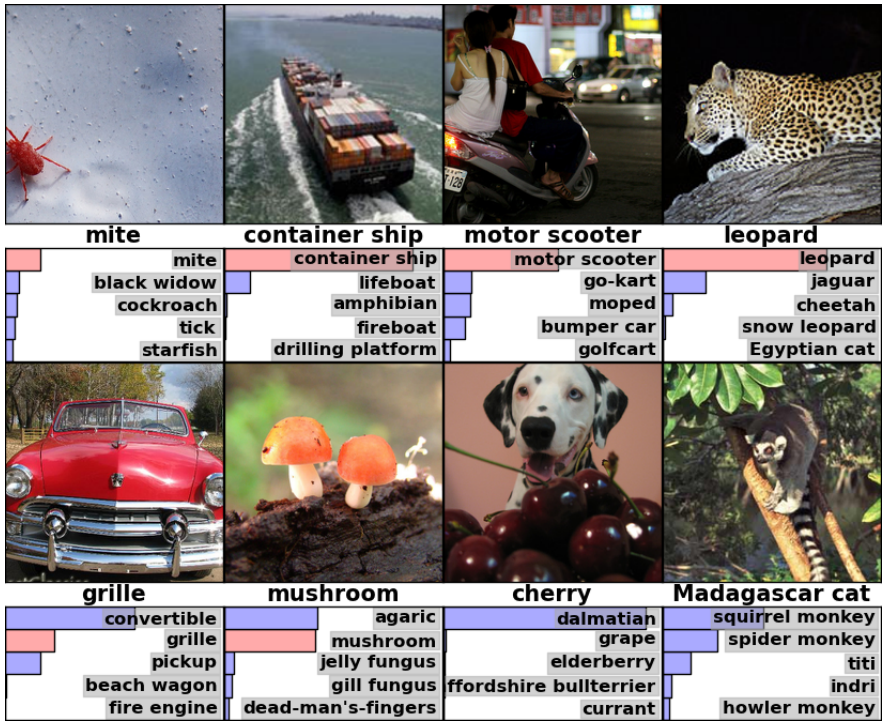
Autonomous Driving

Robots

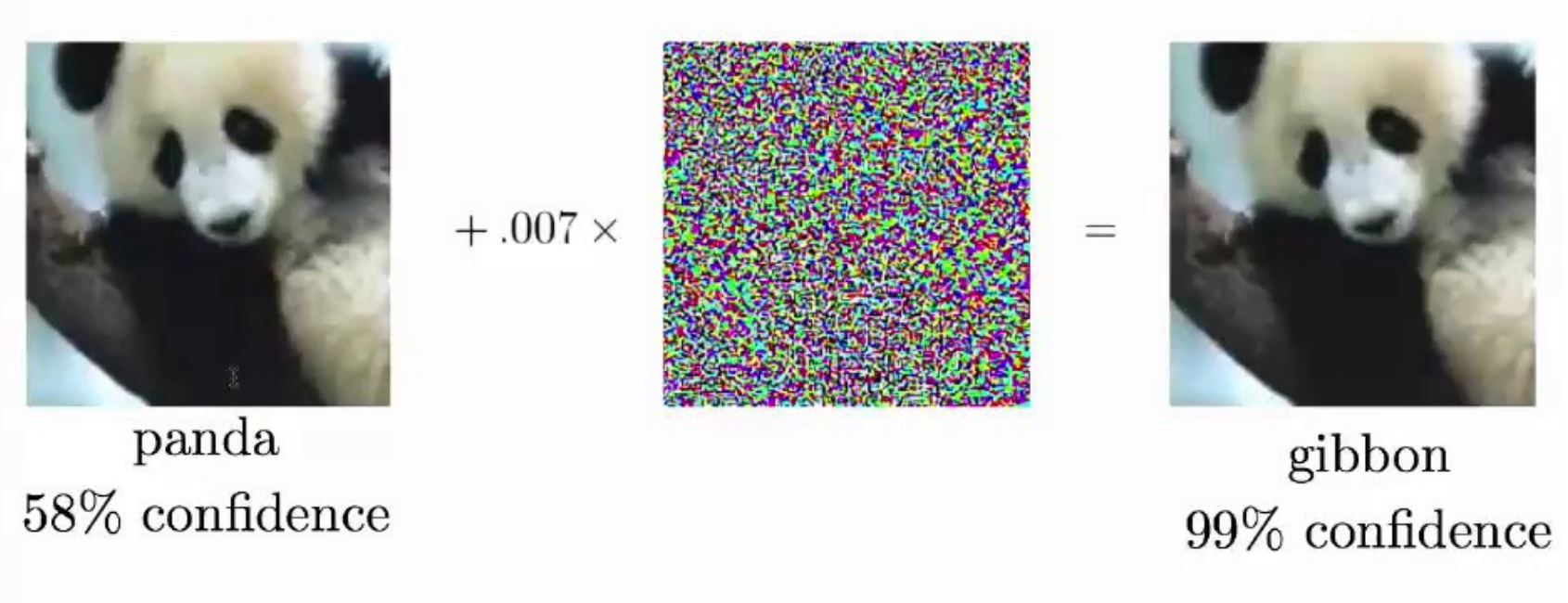


Object detection, face tracking

- Deep neural networks for image classification (Krizhevsky et al. 2012)



Weakness of AIs



- Explaining and Harnessing Adversarial Examples. Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy. ICLR 2015. https://arxiv.org/pdf/1412.6572v3.pdf

Artsy AI

- **Project Magenta** from Google Brain
- Goal: create algorithms that can generate art and music

https://magenta.tensorflow.org

Aiva Technologies

- AIVA: Artificial Intelligence Virtual Artist (http://www.aiva.ai/)
  - Composes convincing classical scores for movies with some guidance from humans

Game Playing

- Chess: IBM's Deep Blue (2006)
- Go: DeepMind's AlphaGo (2016)

Ubiquitous in Tech Companies

Classic AI Problems/Techniques

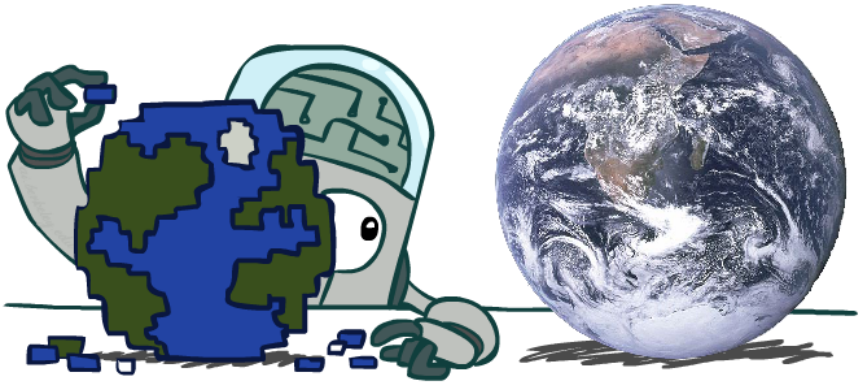
- Search and planning
  - Graph search, goal-informed search
- Constraint Satisfaction Problems
- Probabilistic modeling
  - MDP, HMMs, Bayes' nets
- Machine learning
  - Classification and regression

Classic AI Problems/Techniques

- Search and planning
  - Graph search, goal-informed search
- Constraint Satisfaction Problems
- Probabilistic modeling
  - MDP, HMMs, Bayes' nets
- Machine learning
  - Classification and regression

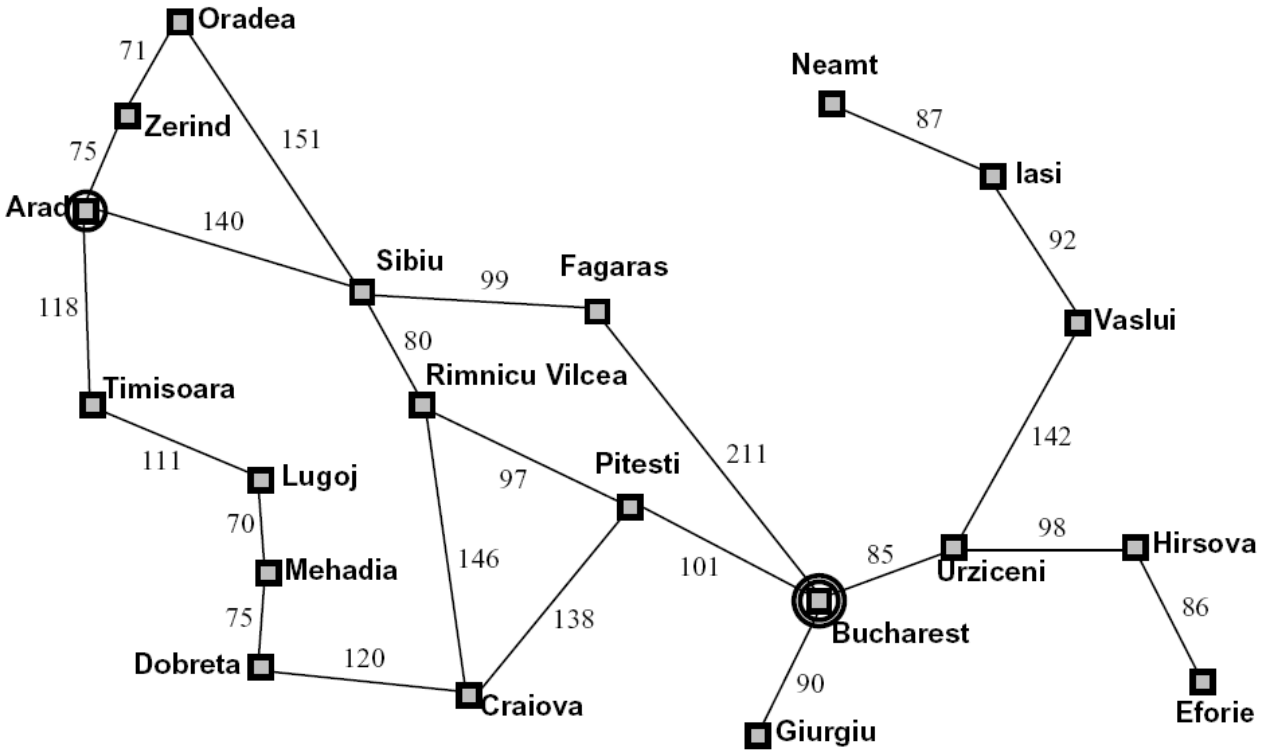
Search Problems

- A search problem consists of:
  - A state space
  - An action set (state-dependent)
  - A transition model
  - Cost (or reward/utility) function
  - Initial state and goal test function
- A solution is a sequence of actions that brings the agent from the initial state to a goal state.



Models approximate the real world!

Example: Travel in Romania



- States: Cities
- Actions:
  Roads connecting adjacent cities
- Transition Model:
  Go to adjacent city
- Cost: distance
- Initial state: Arad
- Goal test:
  - if state == Bucharest

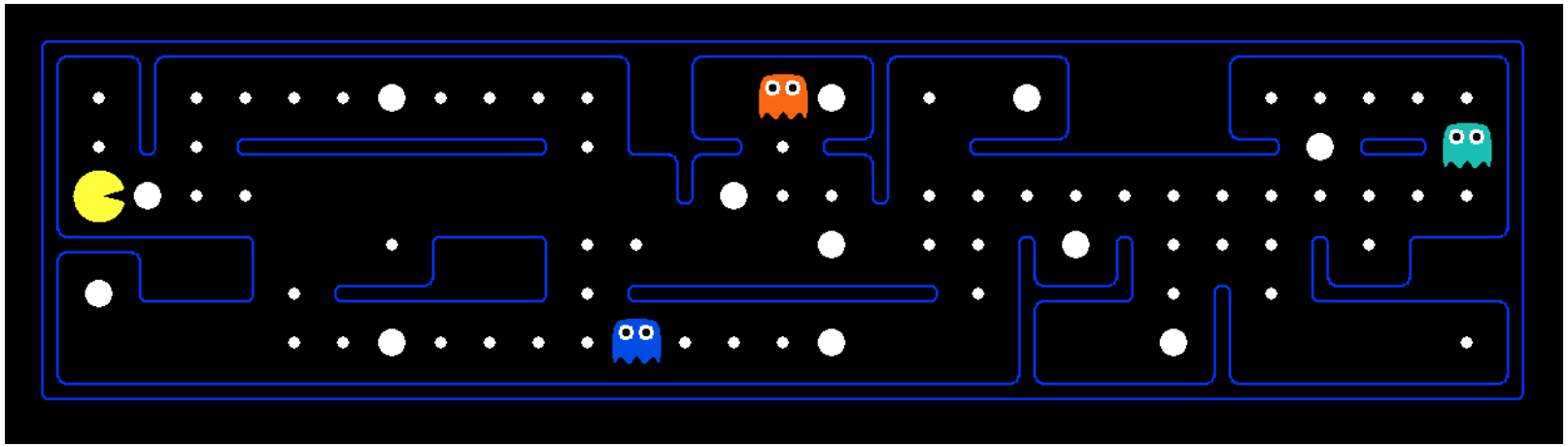
Knuth Root-Factorial Conjecture

- Using a combination of the **square root**, **factorial**, and **floor** operations, we go from 4 to *any positive integer*
- For example:

$$\left\lfloor \sqrt{\sqrt{\left\lfloor \sqrt{\sqrt{\sqrt{\sqrt{(4!)!}}}} \right\rfloor !}} \right\rfloor = 3$$

- **States**: positive reals
- **Actions**: sqrt, floor, ! (integers only)
- **Transitions**: function def., **Goal**: integer equality test

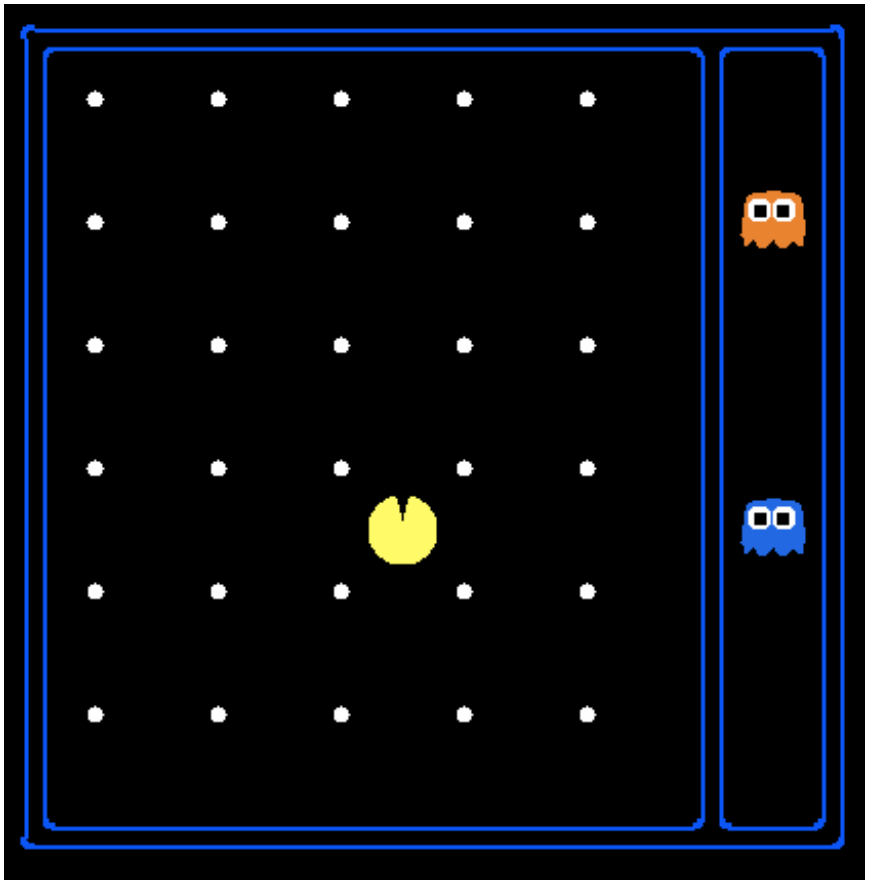
Pac-Man



The **world state** includes everything.

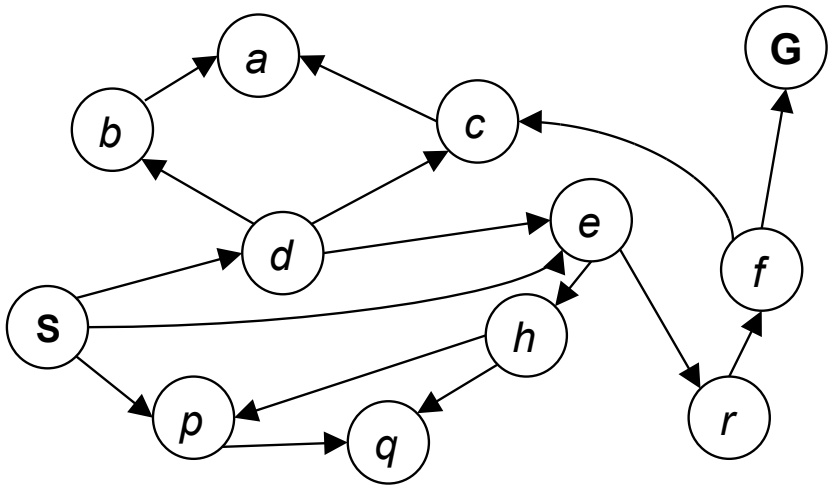
Pac-Man

- Problem: Eat-all-dots
  - States: (x,y,dots boolean)
  - Actions: NSEW
  - Transition: Update location, dot boolean
  - Goal test: all dots are 0



State Space Graphs

- State space graph: A mathematical representation of a domain
  - Nodes are (abstracted) world configurations
  - Edges are (abstracted) actions
  - Goal tests identify membership in the subset of goal nodes
- Drawback: cannot solve large problems — the entire graph is too large

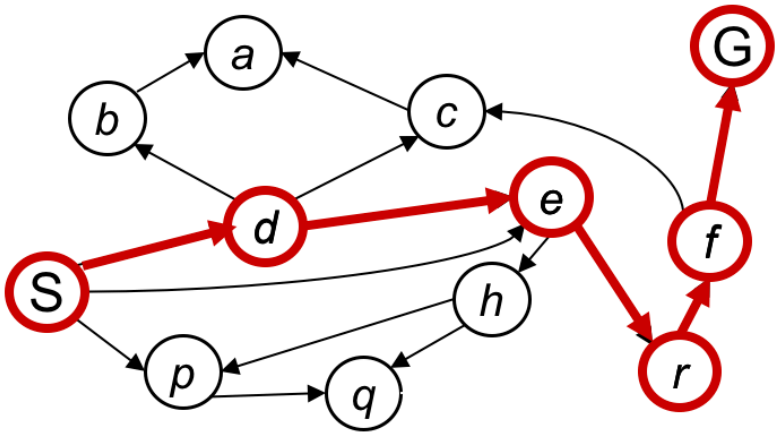


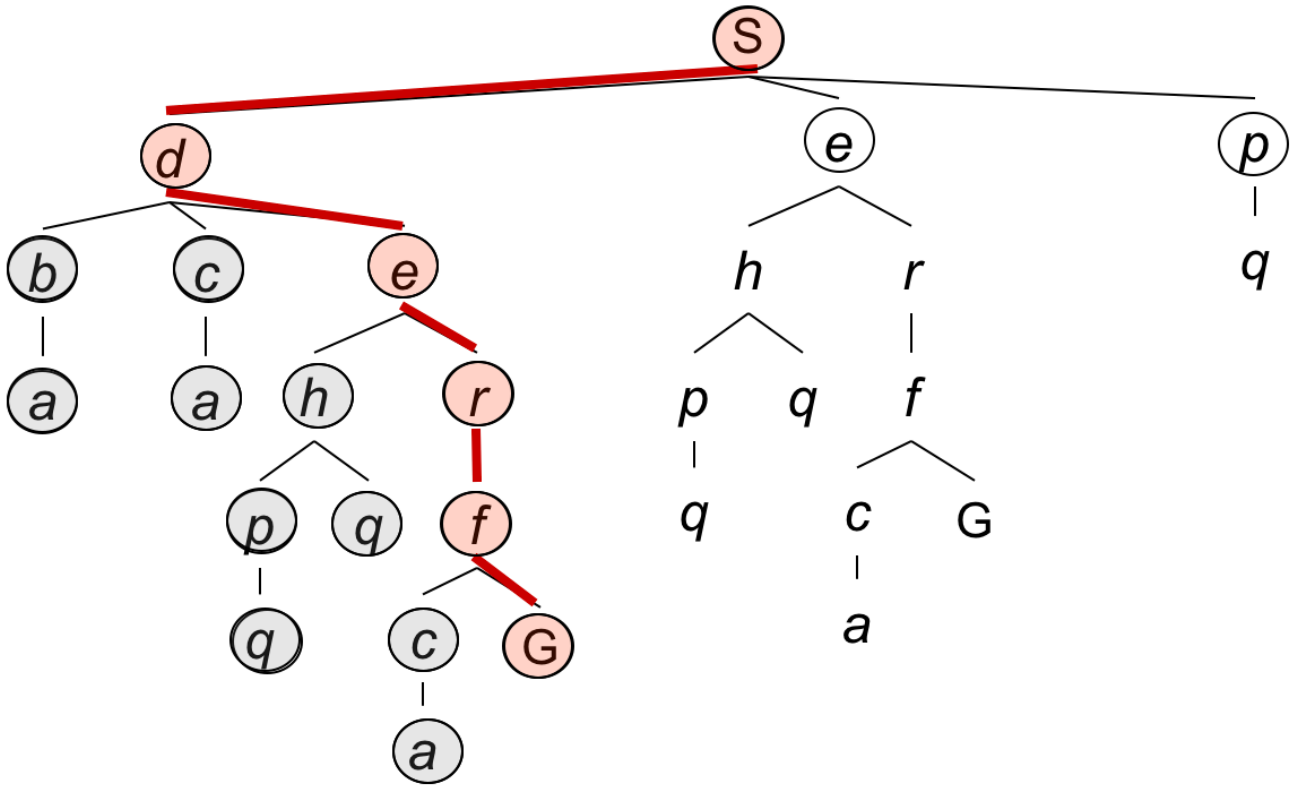
Tree Search Algorithm

- Examples
  - Breath first
  - Depth first

Depth-First Search (DFS)

Strategy: expand the deepest node first

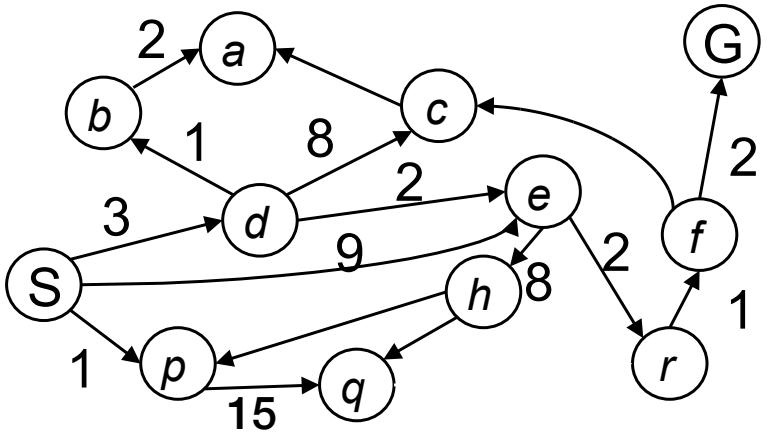


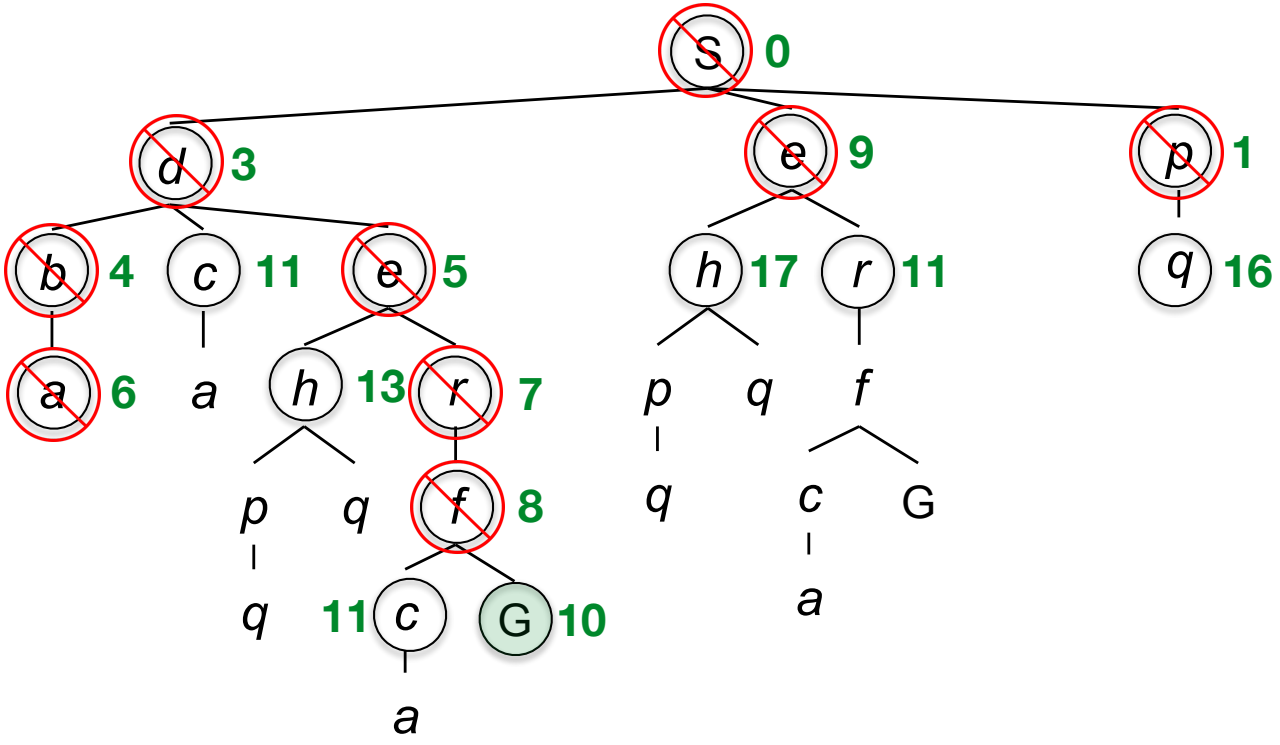


Uniform Cost Search (UCS)

Strategy: expand the node with cheapest total cost

Key idea: use cost information thus far

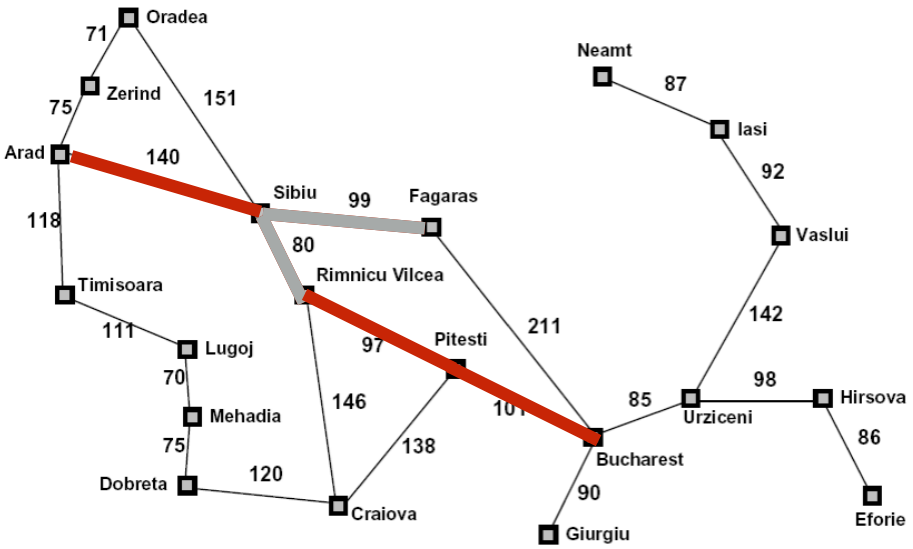




A* Search: combining past and (estimated) future costs

- UCS: prioritize *g(x) = total cost from start*
- A*: prioritize costs in total *f(x) = g(x) + h(x)*
  - *h(x) is **estimated** future costs to reach the goal*

A* Search of Shortest Path



Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

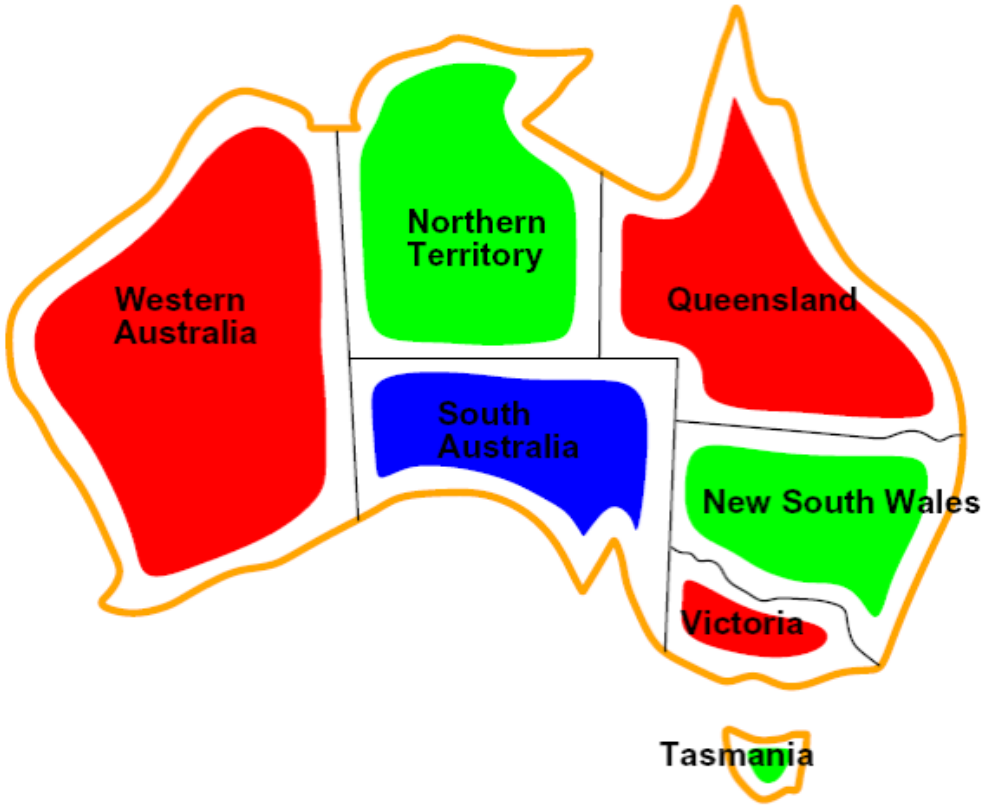
(a) The initial state

366=0+366

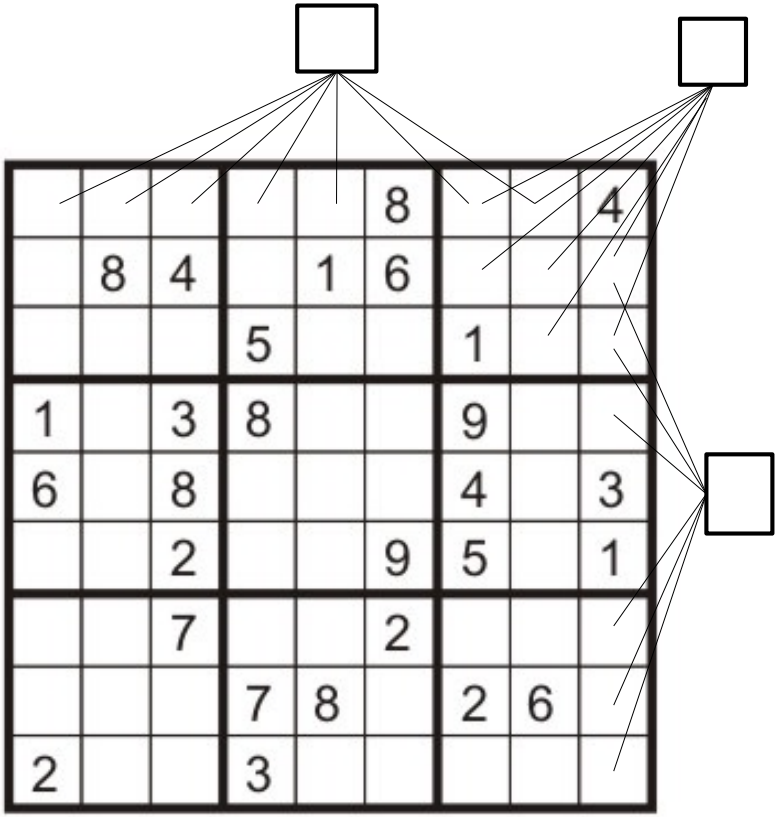
Classic AI Problems/Techniques

- Search and planning
  - Graph search, goal-informed search
- Constraint Satisfaction Problems (CSPs)
- Probabilistic modeling
  - MDP, HMMs, Bayes' nets
- Machine learning
  - Classification and regression

CSP Example: Map Coloring



Example: Sudoku



- Variables:
  - Each (open) square
- Domains:
  - {1,2,…,9}
- Constraints:

9-way *alldiff* for each column

9-way *alldiff* for each row

9-way *alldiff* for each region

CSP as a Search Problem

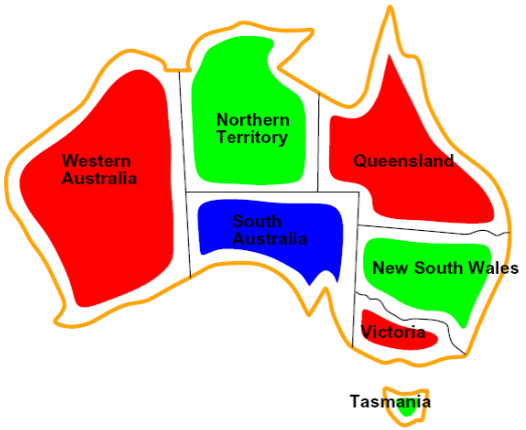
- Variables: WA, NT, Q, NSW, V, SA, T
- Domains: $D = \{red, green, blue\}$
- Constraints: adjacent regions must have different colors

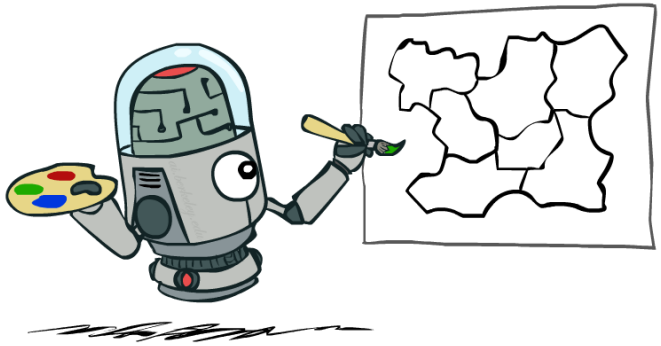
  Implicit: $WA \neq NT$

  Explicit: $(WA, NT) \in \{(red, green), (red, blue), \ldots\}$

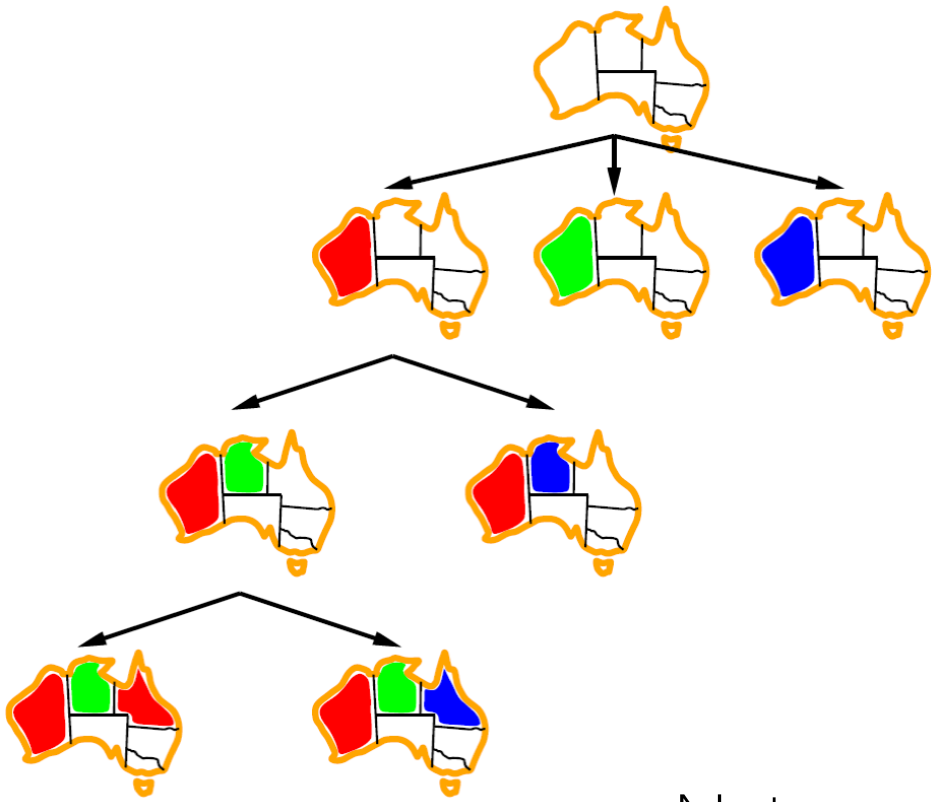
- Solutions are complete assignments satisfying all constraints, e.g., the colored map above

  {WA=red, NT=green, Q=red, NSW=green, V=red, SA=blue, T=green}





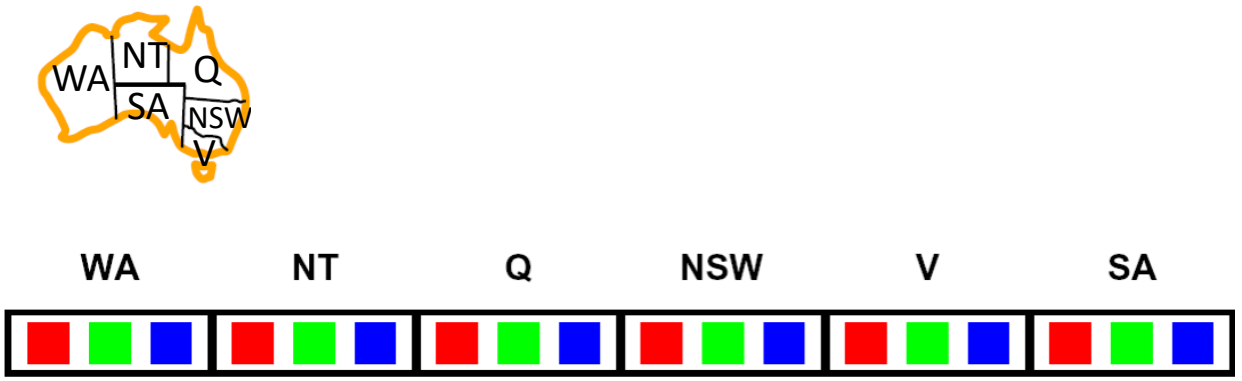
Backtracking Search for CSPs



Note: variables are picked based on fixed order

(Clever) Search with Forward Checking

- Forward checking: Cross off any value for unassigned variables that violates a constraint with the newly assigned variable



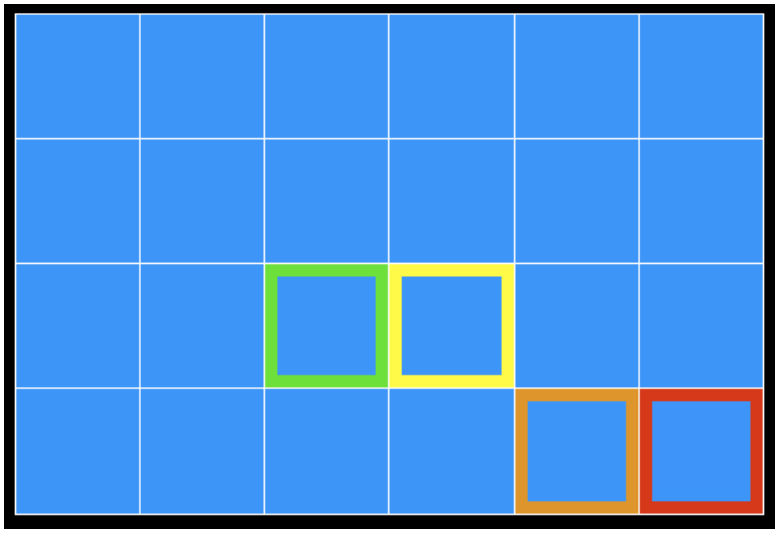
Many other techniques for accelerating this search process

Classic AI Problems/Techniques

- Search and planning
  - Graph search, goal-informed search
- Constraint Satisfaction Problems
- Probabilistic modeling
  - MDP, HMMs, Bayes' nets
- Machine learning
  - Classification and regression

The Need of Probabilistic Reasoning

- A ghost is in the grid somewhere
- Sensor readings tell how close a square is to the ghost
  - On the ghost: red
  - 1 or 2 away: orange
  - 3 or 4 away: yellow
  - 5+ away: green
- Sensors are noisy, but we know *P(Color | Distance)*



Ghostbusters problem

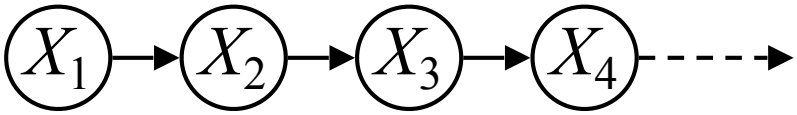
P(yellow | 3)	P(green | 3)	P(orange | 3)	P(red | 3)
0.5	0.3	0.15	0.05

Uncertainty

- Two types of random variables:
  - Observed variables (evidence): Agent knows certain things about the state of the world (e.g., sensor readings)
  - Unobserved (hidden) variables: Agent needs to reason about other aspects (e.g., how far is the ghost)
- Probabilistic model: We know some relation between the known variables and unknown variables
- *Probabilistic reasoning gives us a framework for managing our beliefs and knowledge*

Sequential Probabilistic Models

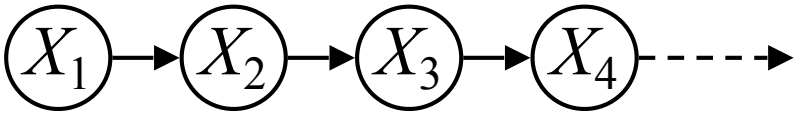
- Value of X at a given time is called the state



- Parameters: initial state + transition probabilities/dynamics, specifying how the state evolves over time

Markov Chains

- Value of *X* at a given time is called the state



Independent

- Assuming that

$$X_3 \perp\!\!\!\perp X_1 \mid X_2 \quad \text{and} \quad X_4 \perp\!\!\!\perp X_1, X_2 \mid X_3$$

- From the chain rule, every joint distribution over can be written as:

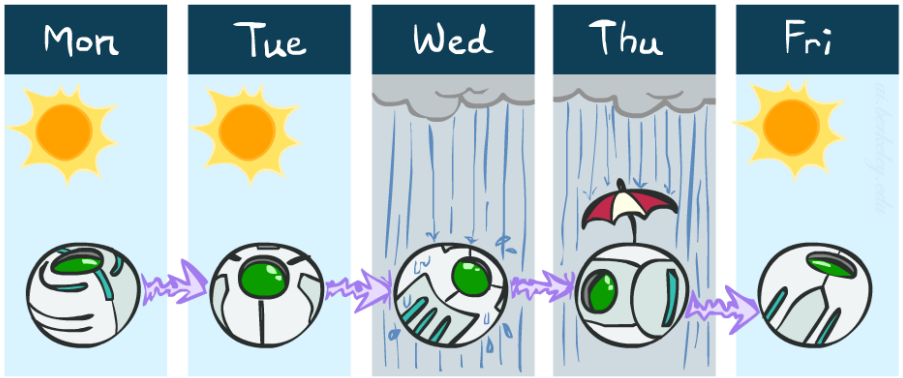
$$P(X_1, X_2, X_3, X_4) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)P(X_4|X_1, X_2, X_3)$$

- Conditional independence assumption implies

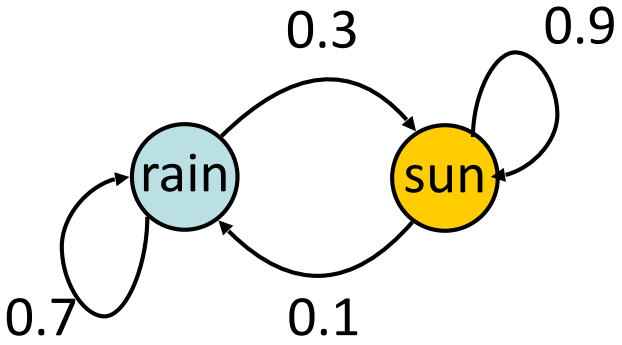
$$P(X_1, X_2, X_3, X_4) = P(X_1)P(X_2|X_1)P(X_3|X_2)P(X_4|X_3)$$

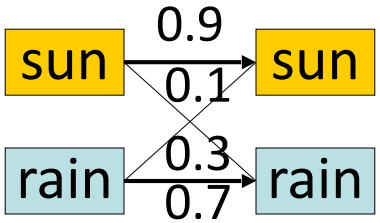
Example Markov Chain: Weather

- Initial distribution: prob1.0 sun
- CPT $P(X_t \mid X_{t-1})$:
- States: $X = \{rain, sun\}$



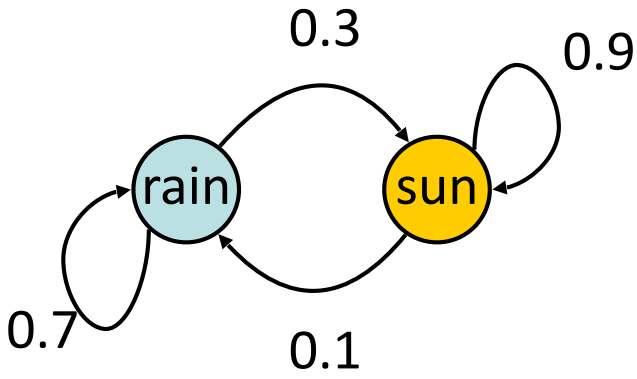
Two ways of representing the same CPT





Example Markov Chain: Weather

- Initial distribution: prob1.0 sun



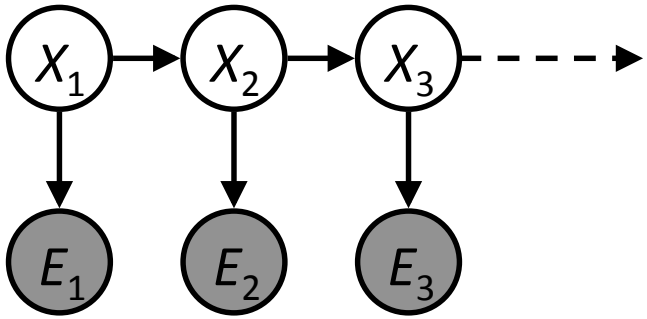
- What is the probability distribution after one step?

$$P(X_2 = \text{sun}) = P(X_2 = \text{sun}|X_1 = \text{sun})P(X_1 = \text{sun}) + P(X_2 = \text{sun}|X_1 = \text{rain})P(X_1 = \text{rain})$$

$$0.9 \cdot 1.0 + 0.3 \cdot 0.0 = 0.9$$

Hidden Markov Models

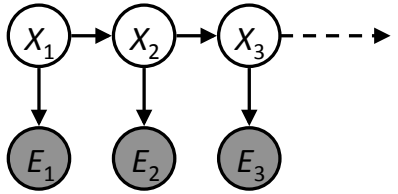
Hidden Markov Models



$P(E|X)$

X	E	P
rain	umbrella	0.9
rain	no umbrella	0.1
sun	umbrella	0.2
sun	no umbrella	0.8

Hidden Markov Models

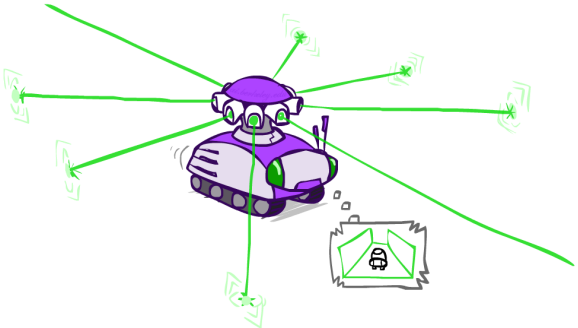


- Input (what we know):
  - Prior over starting states: $P(X_1)$
  - Underlying Markov state dynamics: $P(X_i|X_{i-1})$
  - Emission model (i.e. sensor model): $P(E_i|X_i)$
  - Usually, also emissions E1,…,En

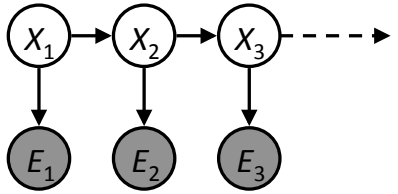
- Output (what we want to do inference to estimate):

$$X_1, \ldots, X_n$$

- Example: robot navigating with imperfect sensors



Hidden Markov Models



- A useful model for environments where we cannot directly observe the state we care, but can observe relevant evidences (e.g., from imperfect sensors)

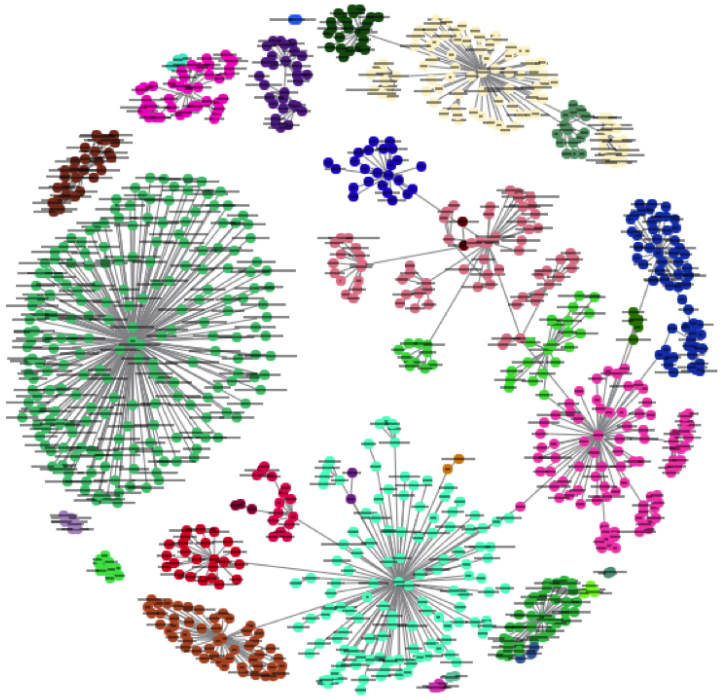
- Major tasks in HMM — infer hidden states from observed evidences
  - Filtering: the task of tracking a *belief distribution* $B_t(X) = P_t(X_t \mid e_1, \ldots, e_t)$

Classic AI Problems/Techniques

- Search and planning
  - Graph search, goal-informed search
- Constraint Satisfaction Problems
- Probabilistic modeling
  - MDP, HMMs, Bayes' nets
- Machine learning
  - Clustering, Classification and regression

Clustering

- Clustering —detect community patterns in unlabeled data
  - E.g. detect communities on FB
  - E.g. group news or movies
  - "Unsupervised" learning

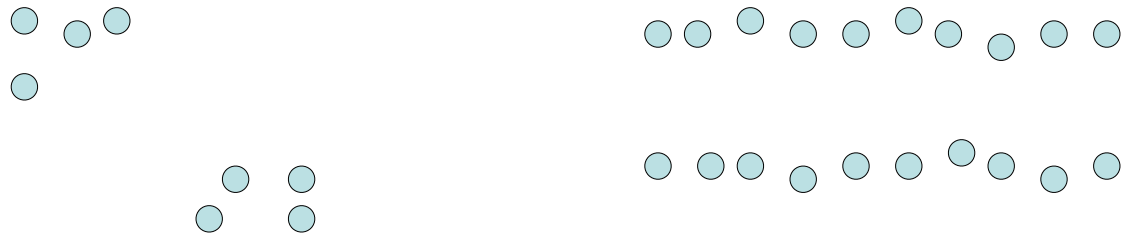


Clustering Examples

- Group customers based on their purchase histories
- Group news into categories
- Identify crime localities
- Identify communities on social networks
- Group movies/tv shows based on viewing histories
- Identify product groups based on the sets of customers who purchased them
- . . . . . .

Clustering

- Basic idea: group similar instances together
- Example: 2D point patterns

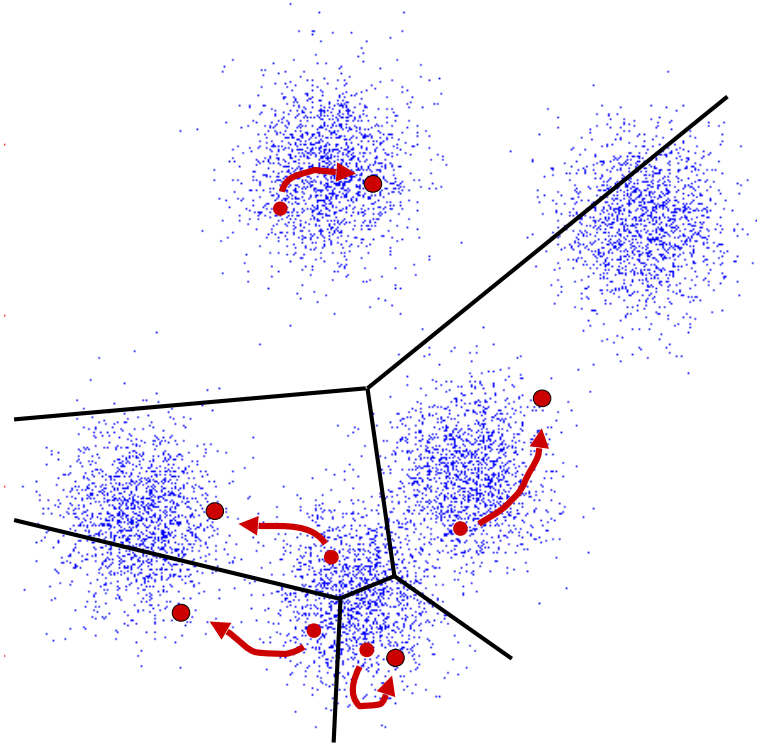


- What could “similar” mean?
  - The canonic option: small (squared) Euclidean distance

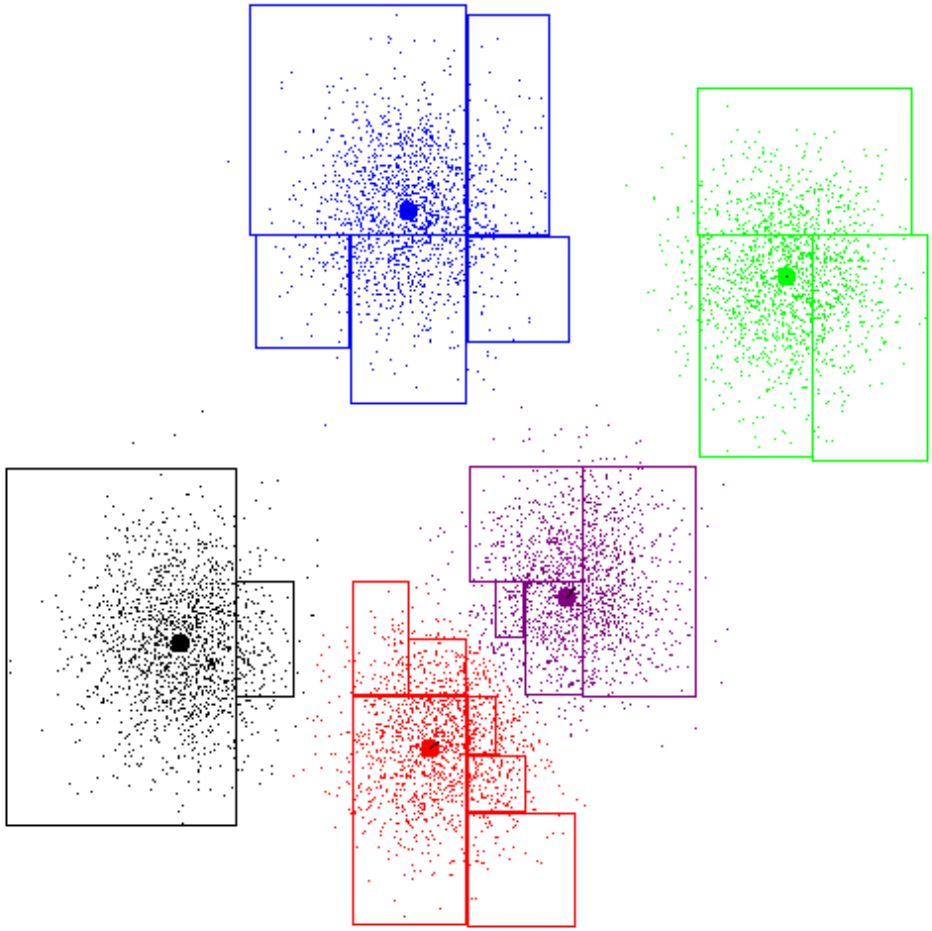
$$\text{dist}(x, y) = (x - y)^{\mathsf{T}}(x - y) = \sum_i (x_i - y_i)^2$$

The K-Means Algorithm

- An iterative clustering algorithm to cluster points into k sets
  - Initialization: pick *K* random points as cluster centers (means)
  - Each iteration:
    - Assign data instances to closest mean
    - Update each mean to the average of its assigned points
  - Repeat until no points' assignments change



K-Means Example

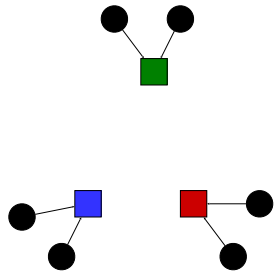


K-Means as Optimization

- Total distance as a function of the means and assignments:

$$\phi(\{x_i\}, \{a_i\}, \{c_k\}) = \sum_i \text{dist}(x_i, c_{a_i})$$

points ↗ assignments ↑ means ↖



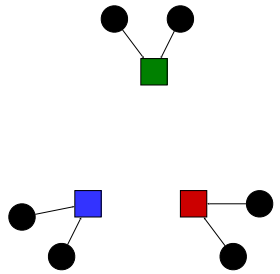
- Clustering seeks to find centers $c_1, \ldots, c_k$ to minimize ϕ

K-Means as Optimization

- Total distance as a function of the means and assignments:

$$\phi(\{x_i\}, \{a_i\}, \{c_k\}) = \sum_i \text{dist}(x_i, c_{a_i})$$

points, assignments, means



- Claim: each iteration of K-means reduces ϕ

- Two phases at each iteration:
  - Update assignments: fix means *c*, change assignments *a*
  - Update means: fix assignments *a*, update means *c*
  - Can prove each phase reduces ϕ