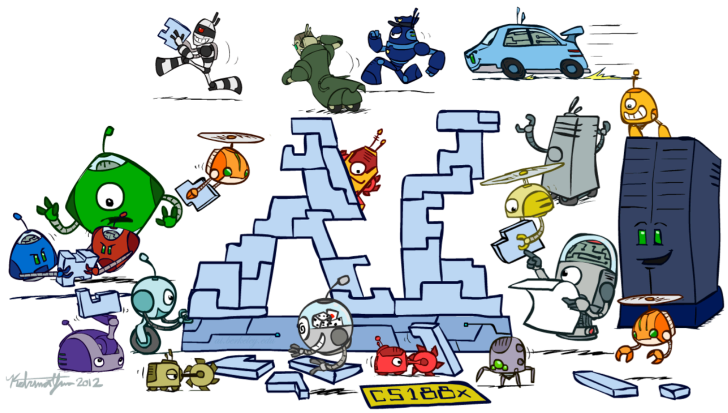
CS 473: Artificial Intelligence
Conclusion



Dan Weld – University of Washington

[Many of these slides were created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley. All CS188 materials are available at http://ai.berkeley.edu.]

Final Exam

- Wed 8:30-10:20
- Closed book
- One 8.5 x 11" sheet of paper notes allowed
- No calculators

2

Studying

- Practice exam & solutions on website
- Review sessions
  - Today 10:30 – my office hour
  - Mon 1:30 – Gagan's office hour
  - Tues – TBD
- Use canvas for questions

3

Exam Topics

- Search
  - Problem spaces
  - BFS, DFS, UCS, A* (tree and graph), local search
  - Completeness and Optimality
  - Heuristics: admissibility and consistency; pattern DBs
- CSPs
  - Constraint graphs, backtracking search
  - Forward checking, AC3 constraint propagation, ordering heuristics
- Games
  - Minimax, Alpha-beta pruning,
  - Expectimax
  - Evaluation Functions
- MDPs
  - Bellman equations
  - Value iteration, policy iteration
- Reinforcement Learning
  - Exploration vs Exploitation
  - Model-based vs. model-free
  - Q-learning
  - Linear value function approx.
- Hidden Markov Models
  - Markov chains, DBNs
  - Forward algorithm
  - Particle Filters
- Bayesian Networks
  - Basic definition, independence (d-sep)
  - Variable elimination
  - Sampling (rejection, importance)
- Learning
  - BN parameters with complete data
  - Search thru space of BN structures
  - Expectation maximization
- Beneficial AI

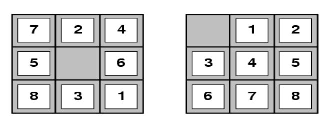
What is intelligence?

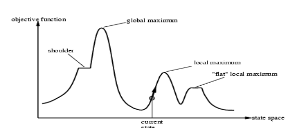
- (bounded) Rationality
  - Agent has a performance measure to optimize
  - Given its state of knowledge
  - Choose optimal action
  - With limited computational resources

- Human-like intelligence/behavior

State-Space Search

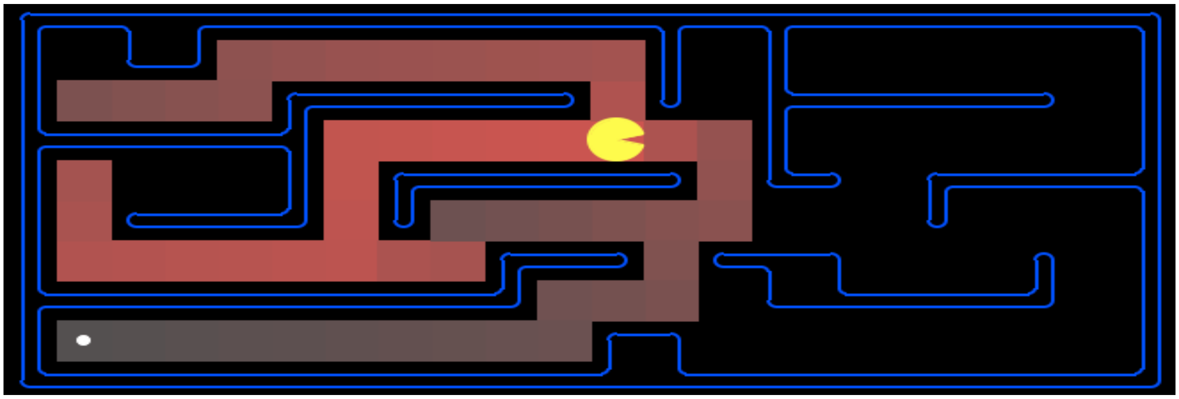
- X as a search problem
  - states, actions, transitions, cost, goal-test
- Types of search
  - uninformed systematic: often slow
    - DFS, BFS, uniform-cost, iterative deepening
  - Heuristic-guided: better
    - Greedy best first, A*
    - Relaxation leads to heuristics
  - Local: fast, fewer guarantees; often local optimal
    - Hill climbing and variations
    - Simulated Annealing: global optimal
  - (Local) Beam Search



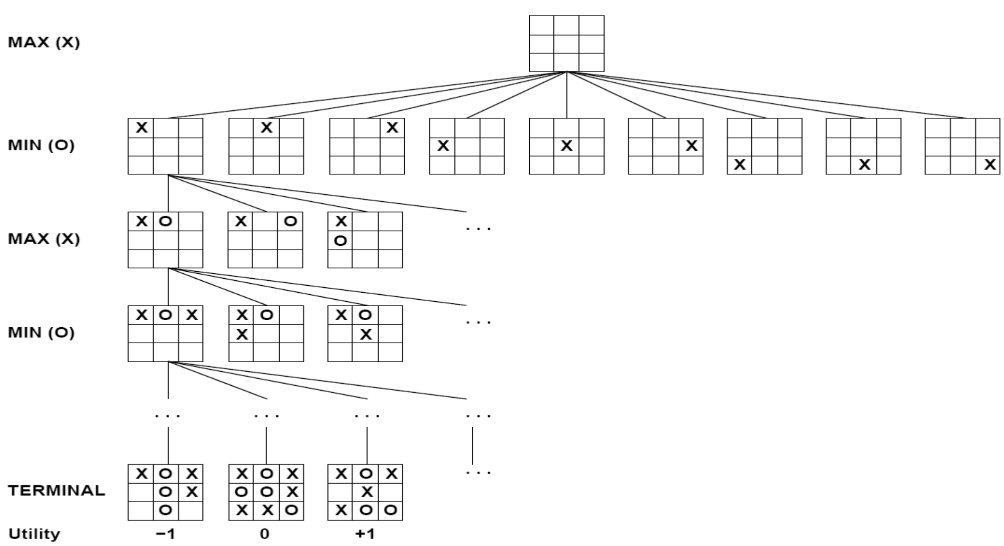


Which Algorithm?

- A*, Manhattan Heuristic:

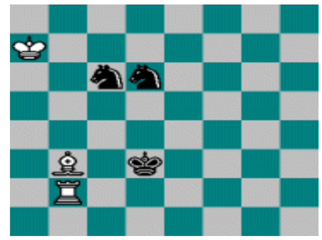


Adversarial Search



Adversarial Search

- AND/OR search space (max, min)
- minimax objective function
- minimax algorithm (~dfs)
  - alpha-beta pruning
- Utility function for partial search
  - Learning utility functions by playing with itself
- Openings/Endgame databases



Knowledge Representation and Reasoning

- Representing: what agent knows

  Propositional logic
  Constraint networks
  HMMs
  Bayesian networks
  …

- Reasoning: what agent can infer

  Search
  Dynamic programming
  Preprocessing to simplify

Knowledge Representation and Reasoning

- Representing: what agent knows
- Reasoning: what agent can infer

{ Propositional logic
Constraint networks
HMMs
Bayesian networks
…

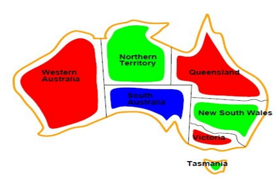
Uncertainty →

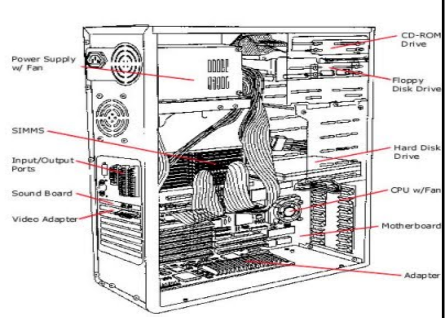
Quantification ↓

Prop Logic Constraint Sat	Bayesian Networks
First-Order Logic	?

Constraint Satisfaction Problems

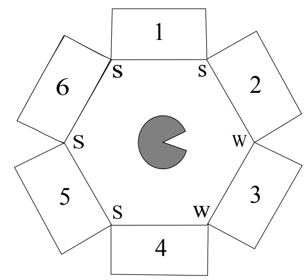
- Representation
  - Variables, Domains, Constraints
- Reasoning:
  - Arc Consistency (k-Consistency)
  - Solving
    - Backtracking search: partial var assignments
      - Heuristics: min remaining values, min conflicts
    - Local search: complete var assignments





Trapped

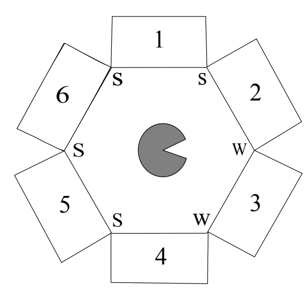
- Pacman is trapped! He is surrounded by mysterious corridors, each of which leads to either a pit (P), a ghost(G), or an exit (E). In order to escape, he needs to figure out which corridors, if any, lead to an exit and freedom, rather than the certain doom of a pit or a ghost.
- The one sign of what lies behind the corridors is the wind: a pit produces a strong breeze (S) and an exit produces a weak breeze (W), while a ghost doesn't produce any breeze at all. Unfortunately, Pacman cannot measure the strength of the breeze at a specific corridor. Instead, he can stand between two adjacent corridors and feel the max of the two breezes. For example, if he stands between a pit and an exit he will sense a strong (S) breeze, while if he stands between an exit and a ghost, he will sense a weak (W) breeze. The measurements for all intersections are shown in the figure below.
- Also, while the total number of exits might be zero, one, or more, Pacman knows that two neighboring squares will not both be exits.



Variables?

Trapped

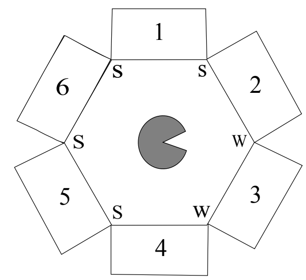
- Pacman is trapped! He is surrounded by mysterious corridors, each of which leads to either a pit (P), a ghost(G), or an exit (E). In order to escape, he needs to figure out which corridors, if any, lead to an exit and freedom, rather than the certain doom of a pit or a ghost.
- The one sign of what lies behind the corridors is the wind: a pit produces a strong breeze (S) and an exit produces a weak breeze (W), while a ghost doesn't produce any breeze at all. Unfortunately, Pacman cannot measure the strength of the breeze at a specific corridor. Instead, he can stand between two adjacent corridors and feel the max of the two breezes. For example, if he stands between a pit and an exit he will sense a strong (S) breeze, while if he stands between an exit and a ghost, he will sense a weak (W) breeze. The measurements for all intersections are shown in the figure below.
- Also, while the total number of exits might be zero, one, or more, Pacman knows that two neighboring squares will not both be exits.



Variables? $X_1, \ldots X_6$
Domains {P, G, E}

Trapped

- A pit produces a strong breeze (S) and an exit produces a weak breeze (W), while a ghost doesn't produce any breeze at all.
- Pacman feels the max of the two breezes.
- the total number of exits might be zero, one, or more,
- two neighboring squares will not both be exits.

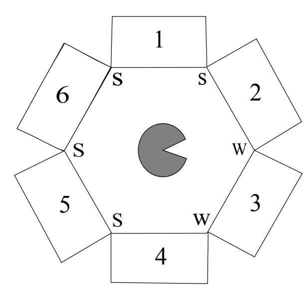


Constraints?

Variables? $X_1, \ldots X_6$
Domains {P, G, E}

Trapped

- A pit produces a strong breeze (S) and an exit produces a weak breeze (W), while a ghost doesn't produce any breeze at all.
- Pacman feels the max of the two breezes.
- the total number of exits might be zero, one, or more,
- two neighboring squares will not both be exits.



Constraints?

$X_1 = P$ or $X_2 = P$
$X_2 = E$ or $X_3 = E$
$X_3 = E$ or $X_4 = E$

$X_4 = P$ or $X_5 = P$
$X_5 = P$ or $X_6 = P$
$X_6 = P$ or $X_1 = P$

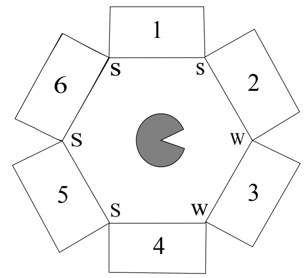
$X_i = E$ nand $X_{i+1|7} = E$

Also! $X_2 =/= P$
$X_3 =/= P$
$X_4 =/= P$

X_1	P	G	E
X_2	P	G	E
X_3	P	G	E
X_4	P	G	E
X_5	P	G	E
X_6	P	G	E

Trapped

- A pit produces a strong breeze (S) and an exit produces a weak breeze (W), while a ghost doesn't produce any breeze at all.
- Pacman feels the max of the two breezes.
- the total number of exits might be zero, one, or more,
- two neighboring squares will not both be exits.



Constraints?

$X_1 = P$ or $X_2 = P$
$X_2 = E$ or $X_3 = E$
$X_3 = E$ or $X_4 = E$

$X_4 = P$ or $X_5 = P$
$X_5 = P$ or $X_6 = P$
$X_6 = P$ or $X_1 = P$

$X_i = E$ nand $X_{i+1|7} = E$

Also! X_2 =/= P
X_3 =/= P
X_4 =/= P

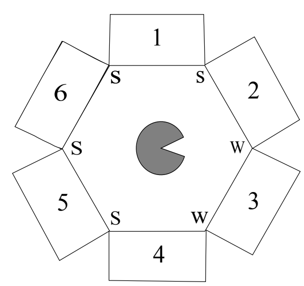
Arc consistent?

X_1	P	G	E
X_2	P	G	E
X_3	P	G	E
X_4	P	G	E
X_5	P	G	E
X_6	P	G	E

17

Trapped

- A pit produces a strong breeze (S) and an exit produces a weak breeze (W), while a ghost doesn't produce any breeze at all.
- Pacman feels the max of the two breezes.
- the total number of exits might be zero, one, or more,
- two neighboring squares will not both be exits.



Constraints?

$X_1 = P$ or $X_2 = P$
$X_2 = E$ or $X_3 = E$
$X_3 = E$ or $X_4 = E$

$X_4 = P$ or $X_5 = P$
$X_5 = P$ or $X_6 = P$
$X_6 = P$ or $X_1 = P$

$X_i = E$ nand $X_{i+1|7} = E$

Also! X_2 =/= P
X_3 =/= P
X_4 =/= P

MRV heuristic?

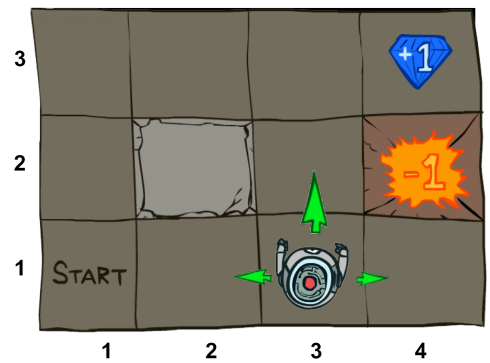
Arc consistent?

X_1	P	~~G~~	~~E~~
X_2	~~P~~	G	E
X_3	~~P~~	G	E
X_4	~~P~~	G	E
X_5	P	~~G~~	~~E~~
X_6	P	G	E

18

KR&R: Markov Decision Process

- Representation
  - states, actions,
  - probabilistic outcomes T ~ P(S' | s, a)
  - Rewards
- Reasoning: V*(s)
  - Value Iteration
    - dynamic programming generalization of expecti-max
  - Policy Iteration



Bellman Equations

$$V^*(s) = \max_a Q^*(s,a)$$

$$Q^*(s,a) = \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s') \right]$$

Value Iteration

Called a "Bellman Backup"

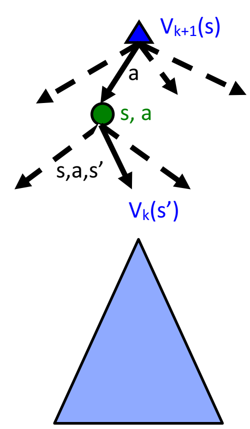
- **Forall s, tialize $V_0(s) = 0$** *no time steps left means an expected reward of zero*
- **Repeat**

  K += 1

  $Q_{k+1}(s, a) = \Sigma_{s'}\ T(s, a, s')\ [\ R(s, a, s') + \gamma\ V_k(s')]$

  $V_{k+1}(s) = \text{Max}_a\ Q_{k+1}(s, a)$

  } do $\forall$ s, a
- **Repeat until $|V_{k+1}(s) - V_k(s)| < \varepsilon$, forall s** *"convergence"*

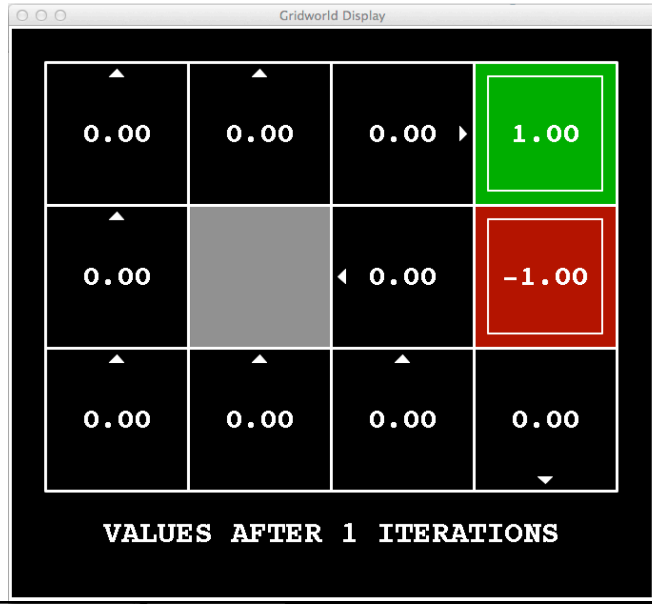


Successive approximation; dynamic programming

k=1

If agent is in 4,3, it only has one legal action: get jewel. It gets a reward and the game is over.
If agent is in the pit, it has only one legal action, die. It gets a penalty and the game is over.

Agent does NOT get a reward for moving INTO 4,3.



Noise = 0.2
Discount = 0.9
Living reward = 0

k=2

0.8 (0 + 0.9*1)
+ 0.1 (0 + 0.9*0)
+ 0.1 (0 + 0.9*0)

Noise = 0.2
Discount = 0.9
Living reward = 0

k=3

Noise = 0.2
Discount = 0.9
Living reward = 0

Policy Iteration

- Let i =0
- Initialize $\pi_i(s)$ to random actions
- Repeat
  - Step 1: Policy evaluation:
    - Initialize k=0; Forall s, $V_0^\pi(s) = 0$
    - Repeat until V^π converges
      - For each state s, $V_{k+1}^{\pi_i}(s) \leftarrow \sum_{s'} T(s, \pi_i(s), s') \left[R(s, \pi_i(s), s') + \gamma V_k^{\pi_i}(s')\right]$
      - Let k += 1
  - Step 2: Policy improvement:
    - For each state, s, $\pi_{i+1}(s) = \arg\max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V^{\pi_i}(s')\right]$
    - If $\pi_i == \pi_{i+1}$ then it's optimal; return it.
    - Else let i += 1

Example

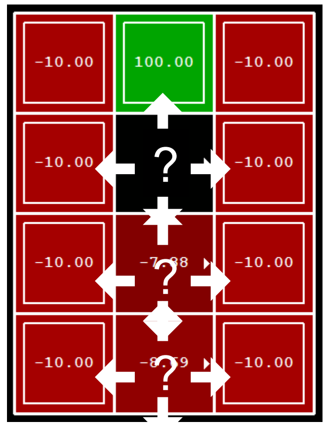
Initialize π_0 to "always go right"

Perform policy evaluation

Perform policy improvement
    Iterate through states

Has policy changed?

Yes! i += 1



Example

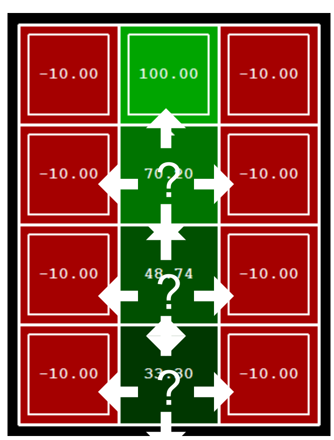
π_1 says "always go up"

Perform policy evaluation

Perform policy improvement
    Iterate through states

Has policy changed?

No! We have the optimal policy



Comparison

- Both value iteration and policy iteration compute the same thing (all optimal values)
- In value iteration:
  - Every iteration updates both the values and (implicitly) the policy
  - We don't track the policy, but taking the max over actions implicitly recomputes it
  - What is the space being searched?
- In policy iteration:
  - We do fewer iterations
  - Each one is slower (must update all V^π and then choose new best π)
  - What is the space being searched?
- Both are dynamic programs for planning in MDPs

Reinforcement Learning

- Model-based vs "model free"
  - I.e. model T(s,a,s), R(s,a,s) explicitly *vs* model Q(s,a)
- Exploration-exploitation tradeoff
  - Epsilon greedy, UCB, …
- Approximating the Q function

"Model Free" RL: Q Learning

- **Forall s, a**
  - **Initialize Q(s, a) = 0**
- **Repeat Forever**

  Where are you? s.

  Choose some action a

  Execute it in real world: *(s, a, r, s')*

  Do update:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + (\alpha)\left[r + \gamma \max_{a'} Q(s',a')\right]$$

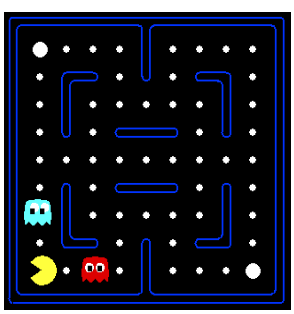
$$\text{difference} = \left[r + \gamma \max_{a'} Q(s',a')\right] - Q(s,a)$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha\,[\text{difference}]$$

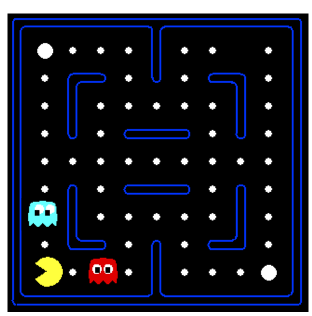
Rewrite as…

Problem: too many states → no generalization

Let's say we discover through experience that this state is bad:



Do we know anything about this one?

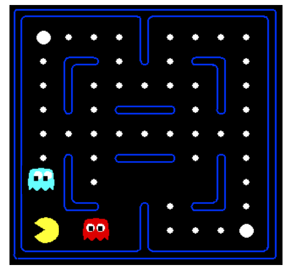


Why isn't this a problem for MDPs & value iteration?

Feature-Based Representations

Soln: describe q-states w/ **vector of features** (aka "properties")

- Features = functions from q-states to R (often 0/1) capturing important properties of the state
- Examples:
  - Distance to closest ghost or dot
  - Number of ghosts
  - Is Pacman in a tunnel? (0/1)
  - Does action move PM closer to ghost?



- Define:

$$Q(s,a) = w_1f_1(s,a) + w_2f_2(s,a) + \ldots + w_nf_n(s,a))$$

Approximate Q-Learning

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$

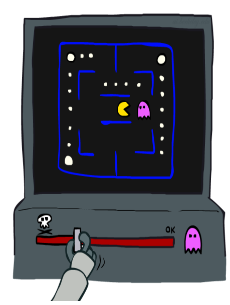
- Q-learning with linear Q-functions:

$$\text{transition } = (s, a, r, s')$$

$$\text{difference} = \left[r + \gamma \max_{a'} Q(s', a')\right] - Q(s,a)$$

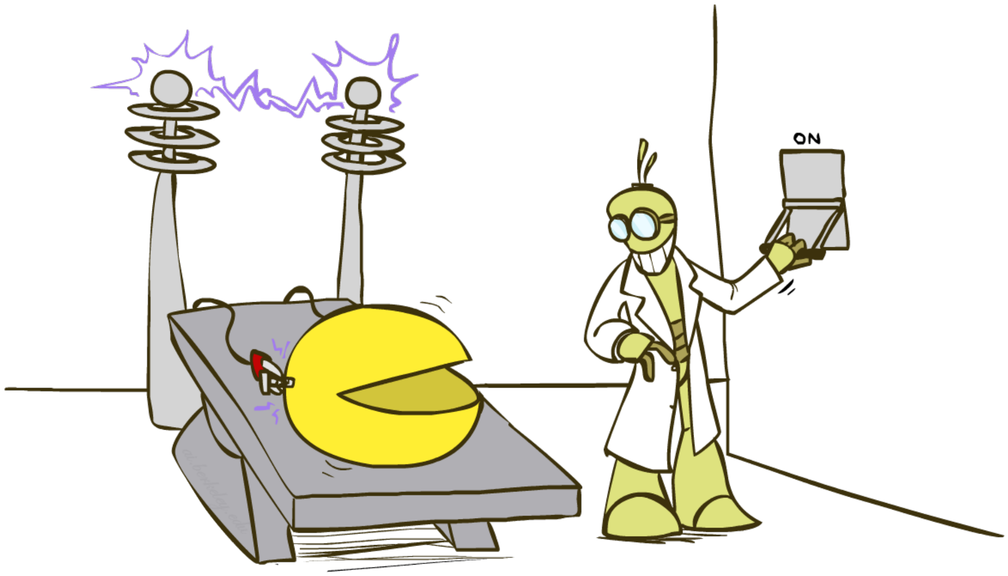
$$Q(s,a) \leftarrow Q(s,a) + \alpha\,[\text{difference}]$$ Old way: Exact Q's

$$w_i \leftarrow w_i + \alpha\,[\text{difference}]\, f_i(s,a)$$ Now: Approximate Q's



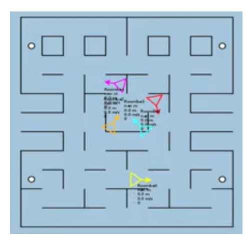
- Intuitive interpretation:
  - Adjust weights of active features
  - E.g., if something unexpectedly bad happens, blame the features that were ***active***: ***disprefer all states with that state's features***

Pac-Man Beyond the Game!



Pacman: Beyond Simulation?

ROOMBA PACMAN



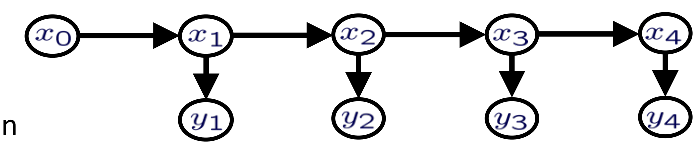
Students at Colorado University: http://pacman.elstonj.com

[VIDEO: Roomba Pacman.mp4]

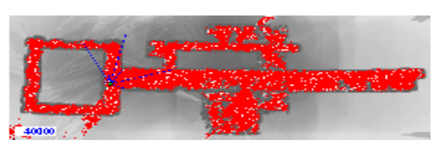
Pacman: Beyond Simulation!

KR&R: Hidden Markov Models

- Representation
  - Simple form of BN
  - Sequence model
  - One hidden state, one observation

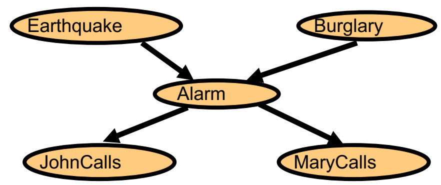


- Reasoning/Search
  - probability of final state: forward algorithm
  - marginal prob of one state: forward-backward
  - most likely state sequence: Viterbi algorithm



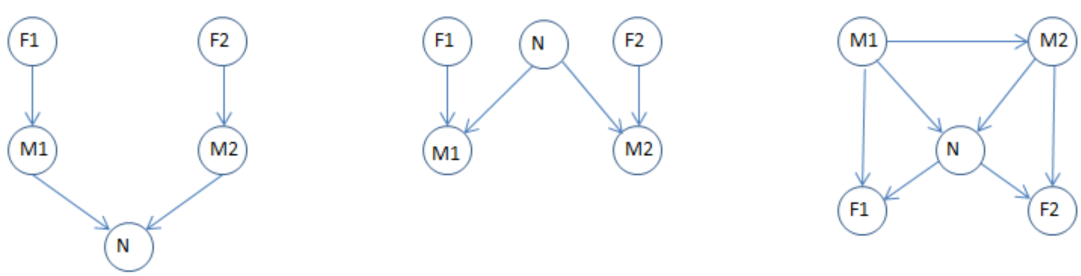
KR&R: Probability

- Representation: Bayesian Networks
  - encode probability distributions compactly
    - by exploiting conditional independences



- Reasoning
  - Exact inference: var elimination
  - Approx inference: sampling based methods
    - rejection sampling, likelihood weighting, MCMC/Gibbs

Problem 10 (12 points) Two astronomers in different parts of the world make measurements, M1 and M2, of the number of stars, N, in some small region of the sky, using their telescopes. Normally, there is a small possibility ε of error by up to one star in each direction. Eash telescope can also (with a much smaller probability *f*) be badly out of focus (events F1 and F2), in which case the scientist will undercount by three or more starts (or if N is less than 3, fail to detect any stars at all). Consider the three networks shown below:



(a) (b) (c)

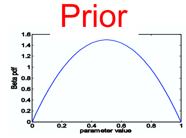
- Which network(s) can represent this information?
  - B,C. (A has N conditionally independent of F1 given M)
- Which is the best?

Bayesian Learning

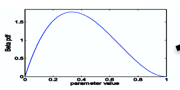
Use Bayes rule:

Data Likelihood

Prior



Posterior



$$P(Y \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid Y)\, P(Y)}{P(\mathbf{X})}$$

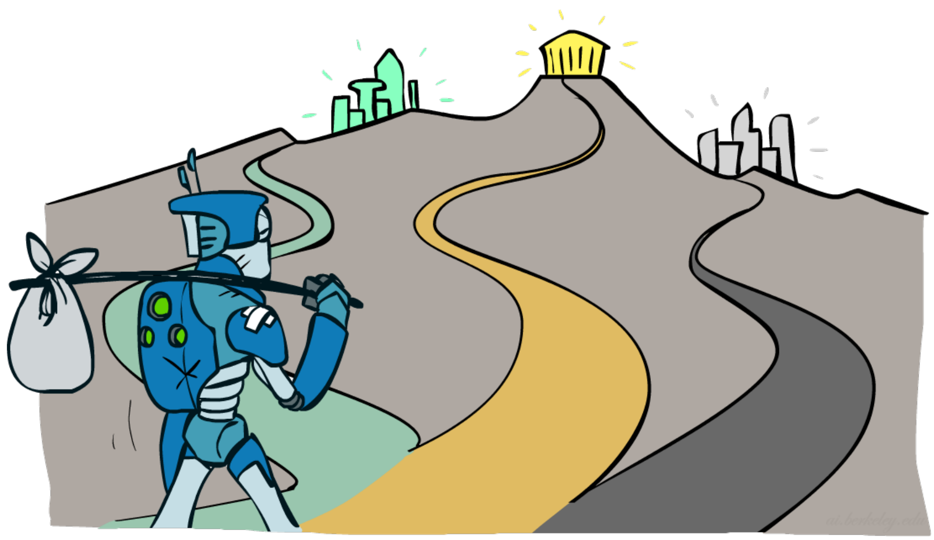
Normalization

Or equivalently: $P(Y \mid \mathbf{X}) \propto P(\mathbf{X} \mid Y)\, P(Y)$

Learning Bayes Networks

- Learning Structure of Bayesian Networks
  - Search thru space of BN structures
- Learning Parameters for a Bayesian Network
  - Fully observable variables
    - Maximum Likelihood (ML), MAP & Bayesian estimation
    - Example: Naïve Bayes for text classification
  - Hidden variables
    - Expectation Maximization (EM)

Where to Go Next?



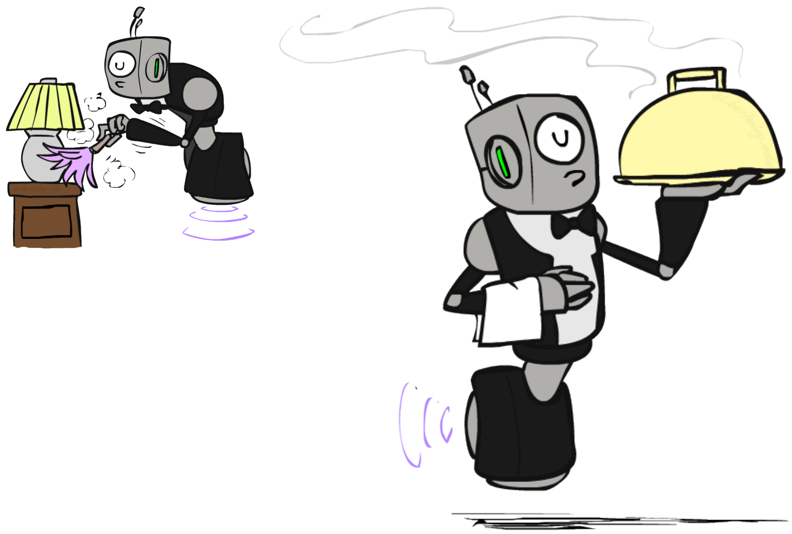
Next…

- CSE 427 – Computational Biology
- CSE 446 – Machine Learning (winter, spring)
- CSE 455 – Computer Vision (winter)
- CSE 490u – Natural Language Processing (winter)
- CSE 481c – Robotics Capstone (spring)
- CSE 481d – Games Capstone (spring) ????
- CSE 481nlp- NLP Capstone (spring)

- CSE 515 – Statistical Methods in CS (winter)
- CSE 547 Machine Learning for Big Data (spring)
- CSE 579 – Intelligent Control (spring) ???

42

Personal Robotics



PR2 (autonomous)

[VIDEO: 5pile_200x.mp4]

[Maitin-Shepard, Cusumano-Towner, Lei, Abbeel, 2010]

Autonomous tying of a knot for previously unseen situations

[VIDEO: knots_apprentice.mp4]

[Schulman, Ho, Lee, Abbeel, 2013]

That’s It!

- Help us out with some course evaluations
- Have a great holiday, and always maximize your expected utilities!