

# Artificial Intelligence

CS 165A

Apr 24, 2022

Instructor: Prof. Yu-Xiang Wang

Today

- Finish games and minimax search
- Midterm review

# Recap: Games and minimax search

- Perfect information, turn-based, Deterministic, zero-sum game
- Formulation as a search problem: MIN and MAX alternatively.
- Alpha/Beta Pruning of the Search Tree
- Early Cut-off search with an heuristic function

# Today

- Finish games and adversarial search:
  - Playing against multiple opponents
  - Exploiting benign opponent with Expectimax
- Midterm review

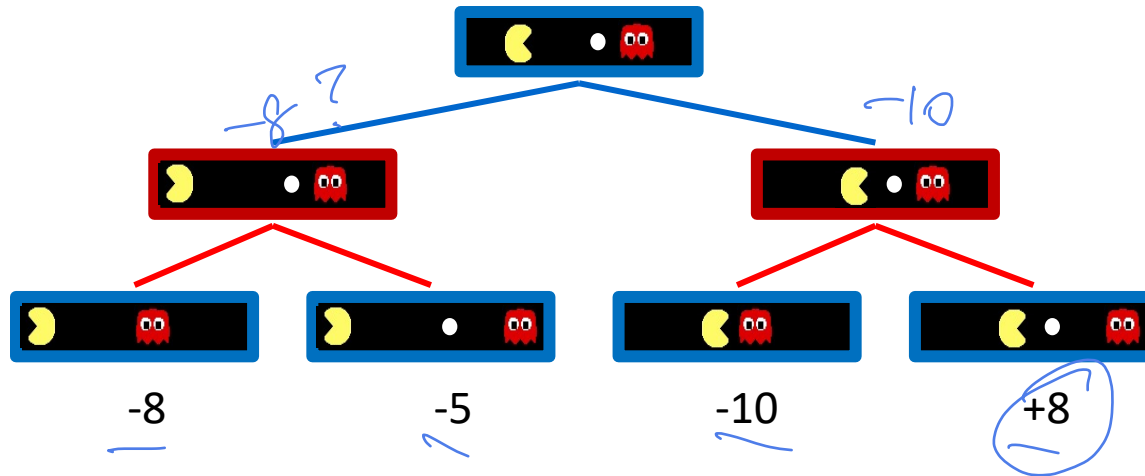
# Small Pacman Example

MAX nodes: under Agent's control

$$V(s) = \max_{s' \in \text{successors}(s)} V(s')$$

MIN nodes: under Opponent's control

$$V(s) = \min_{s' \in \text{successors}(s)} V(s')$$



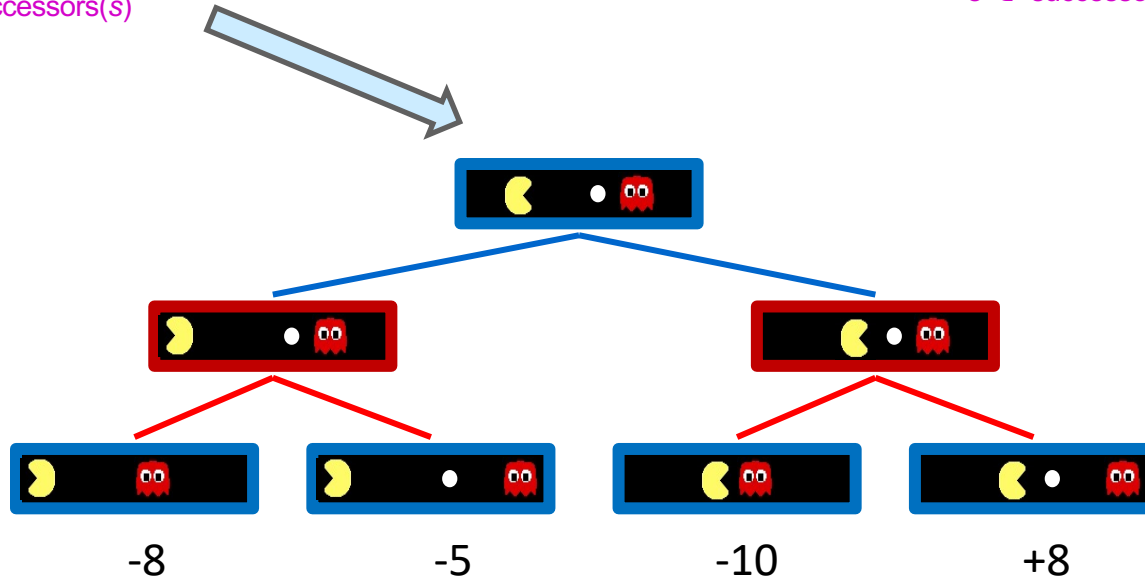
# Small Pacman Example

MAX nodes: under Agent's control

$$V(s) = \max_{s' \in \text{successors}(s)} V(s')$$

MIN nodes: under Opponent's control

$$V(s) = \min_{s' \in \text{successors}(s)} V(s')$$



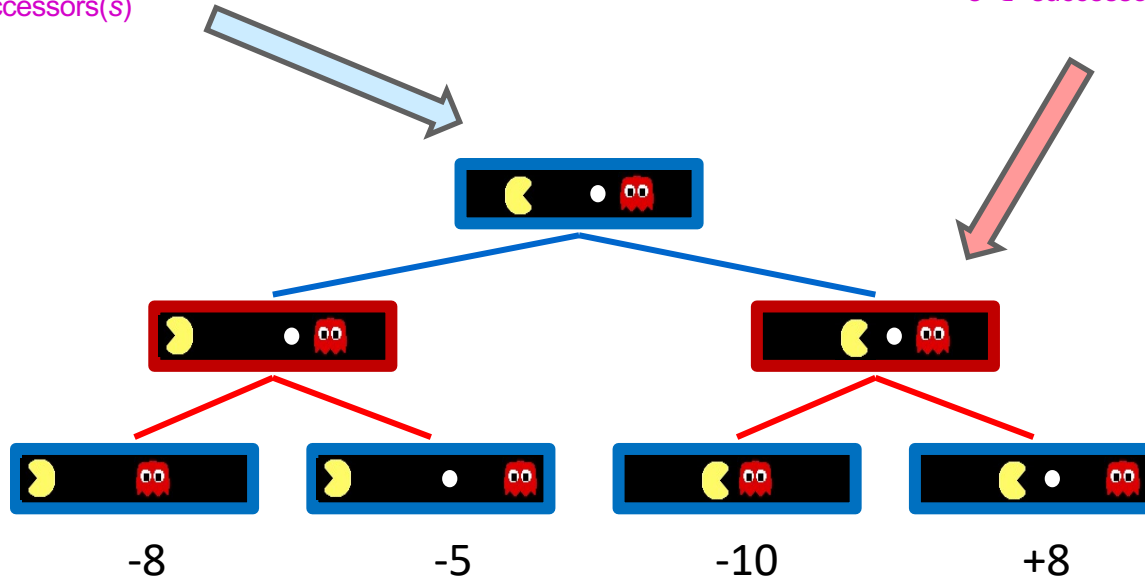
# Small Pacman Example

MAX nodes: under Agent's control

$$V(s) = \max_{s' \in \text{successors}(s)} V(s')$$

MIN nodes: under Opponent's control

$$V(s) = \min_{s' \in \text{successors}(s)} V(s')$$



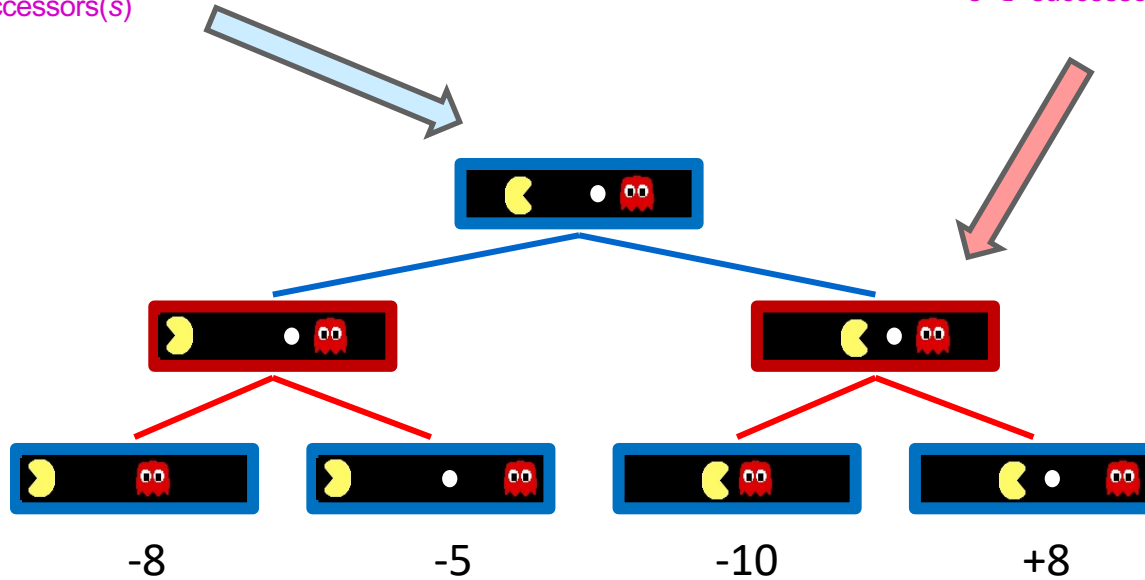
# Small Pacman Example

MAX nodes: under Agent's control

$$V(s) = \max_{s' \in \text{successors}(s)} V(s')$$

MIN nodes: under Opponent's control

$$V(s) = \min_{s' \in \text{successors}(s)} V(s')$$



Terminal States:

$$V(s) = \text{known}$$

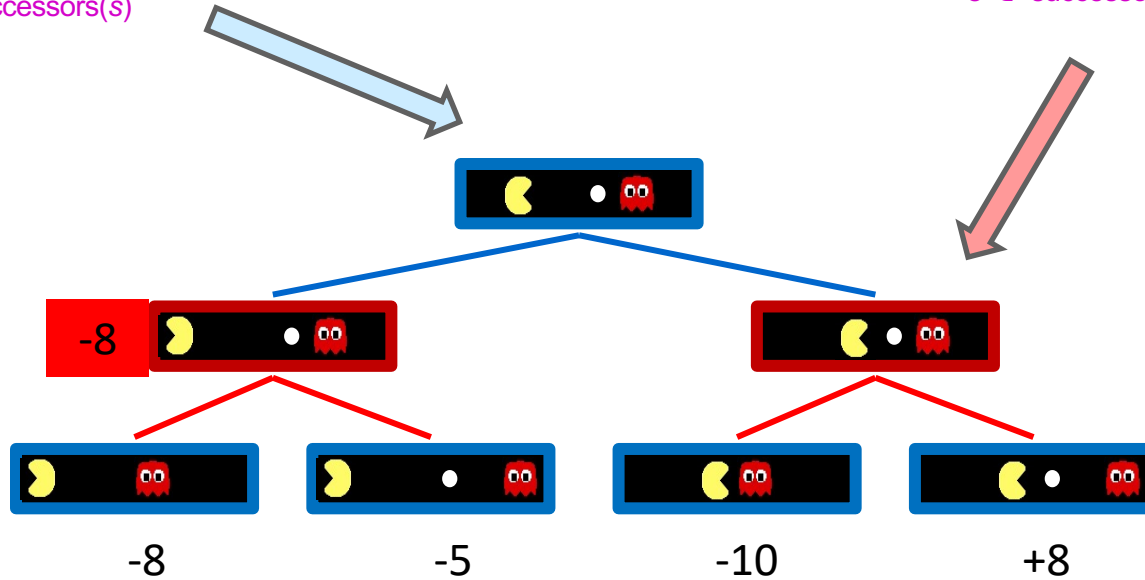
# Small Pacman Example

MAX nodes: under Agent's control

$$V(s) = \max_{s' \in \text{successors}(s)} V(s')$$

MIN nodes: under Opponent's control

$$V(s) = \min_{s' \in \text{successors}(s)} V(s')$$



Terminal States:

$$V(s) = \text{known}$$



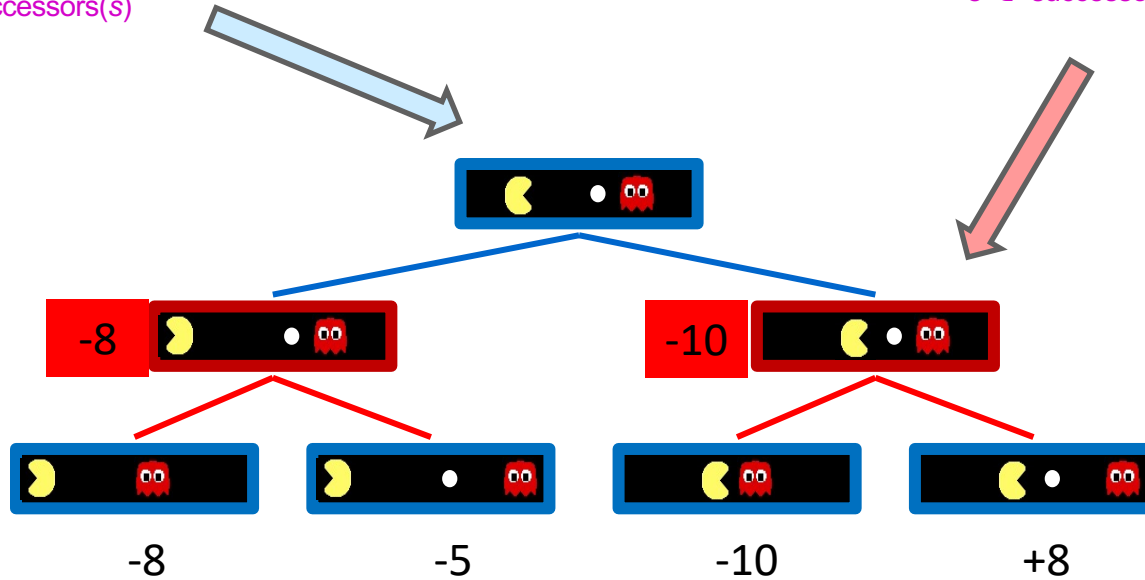
# Small Pacman Example

MAX nodes: under Agent's control

$$V(s) = \max_{s' \in \text{successors}(s)} V(s')$$

MIN nodes: under Opponent's control

$$V(s) = \min_{s' \in \text{successors}(s)} V(s')$$



Terminal States:

$$V(s) = \text{known}$$

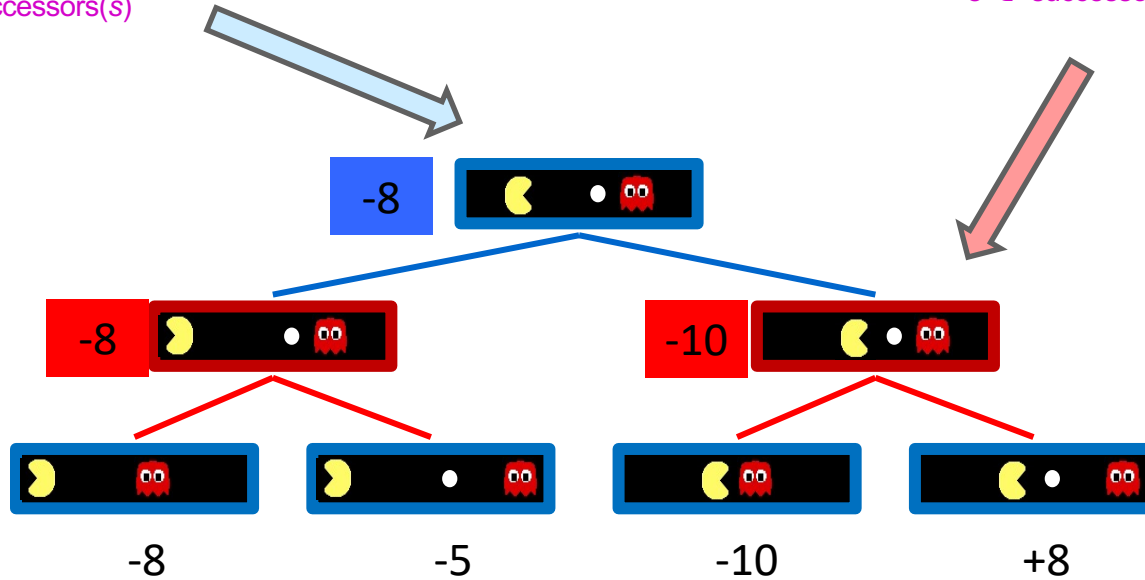
# Small Pacman Example

MAX nodes: under Agent's control

$$V(s) = \max_{s' \in \text{successors}(s)} V(s')$$

MIN nodes: under Opponent's control

$$V(s) = \min_{s' \in \text{successors}(s)} V(s')$$



Terminal States:

$$V(s) = \text{known}$$

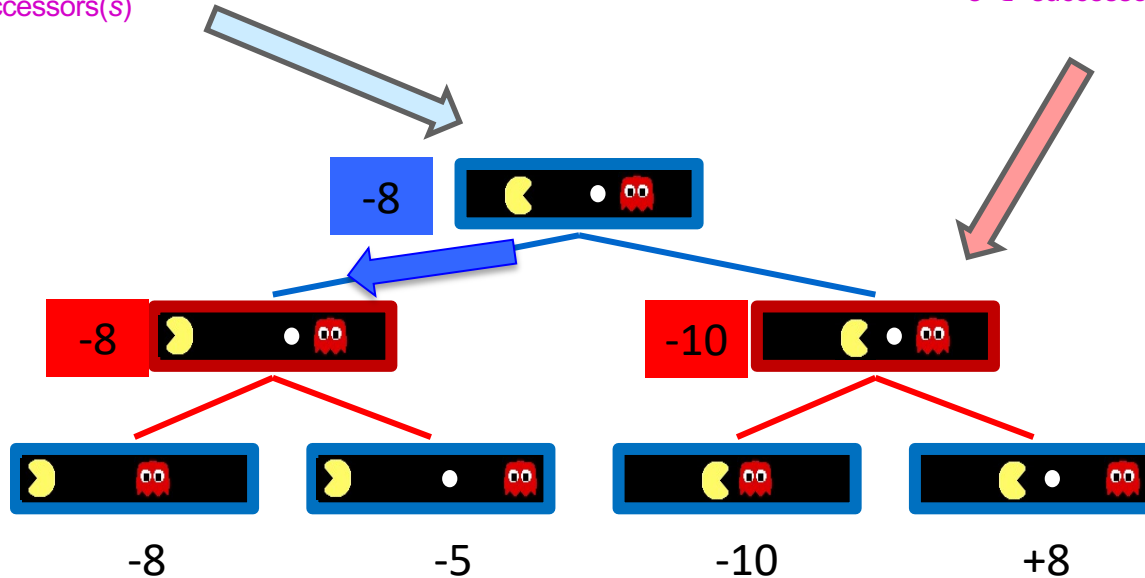
# Small Pacman Example

MAX nodes: under Agent's control

$$V(s) = \max_{s' \in \text{successors}(s)} V(s')$$

MIN nodes: under Opponent's control

$$V(s) = \min_{s' \in \text{successors}(s)} V(s')$$

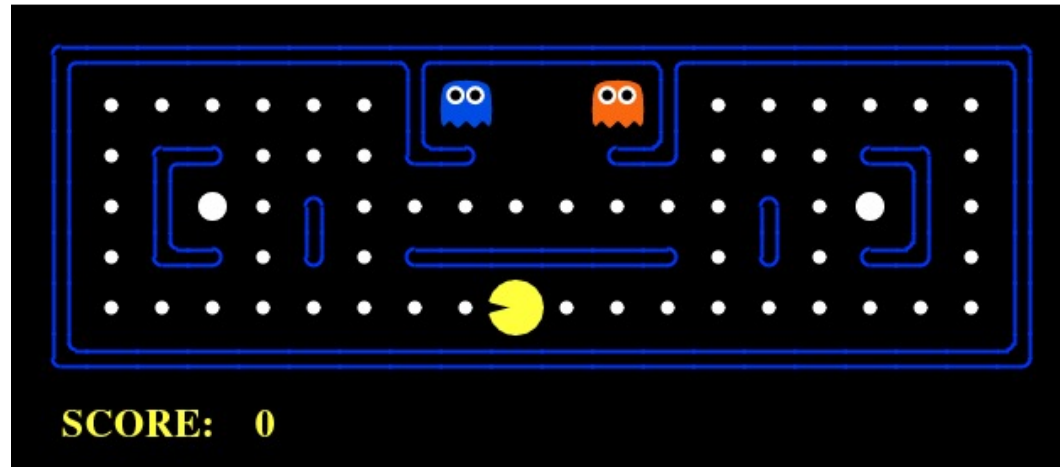


Terminal States:

$$V(s) = \text{known}$$

# Coding Project 2

- PACMAN with ghosts



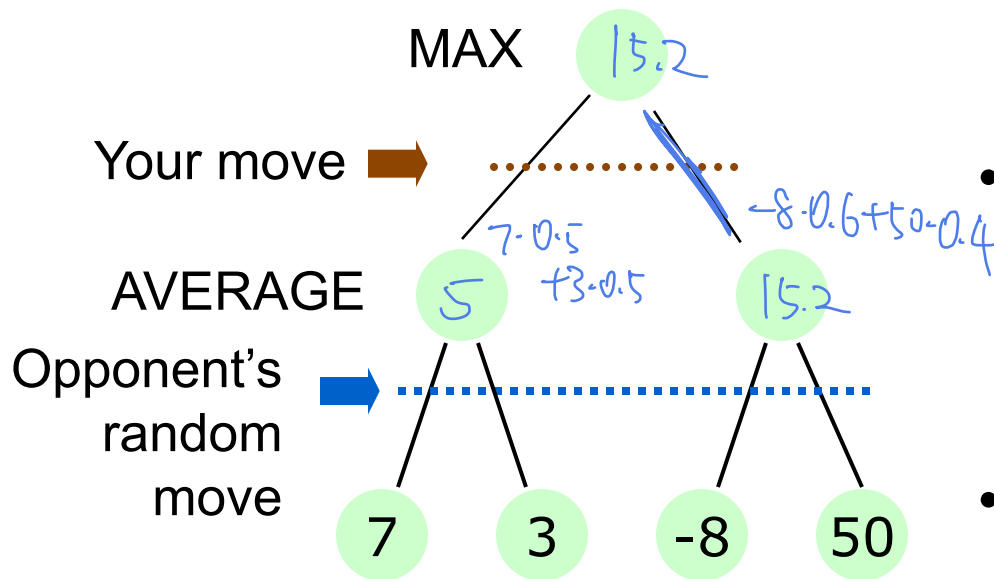
- You will generalize minimax search to multiple opponents in Q7
- You will exploit the benign opponents in Q8

# Expectimax: Playing against a benign opponent

- Sometimes your opponents are not clever.
  - They behave randomly.
  - You can take advantage of that by modeling your opponent.
- Example of game of chance:
  - Slot machines
  - Tetris

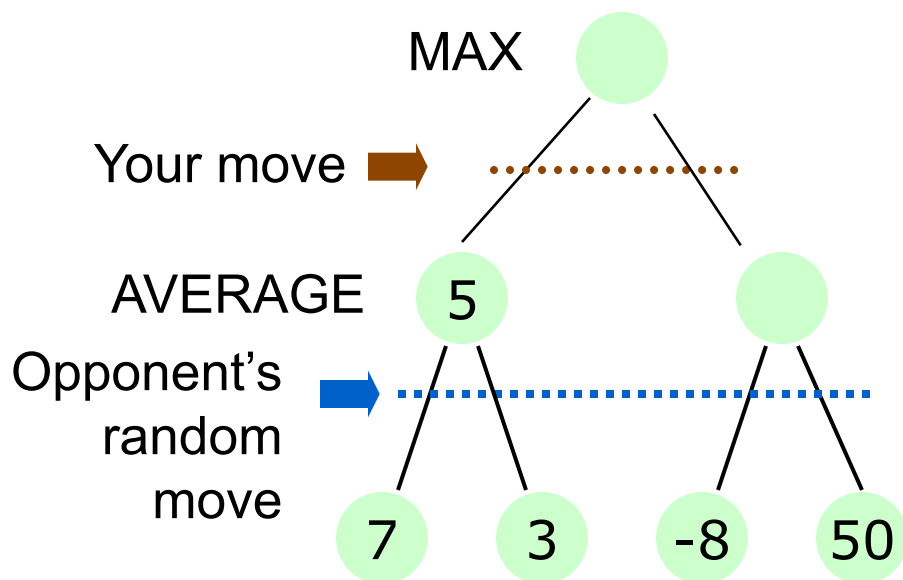
Expecti minimax

# Expectimax example



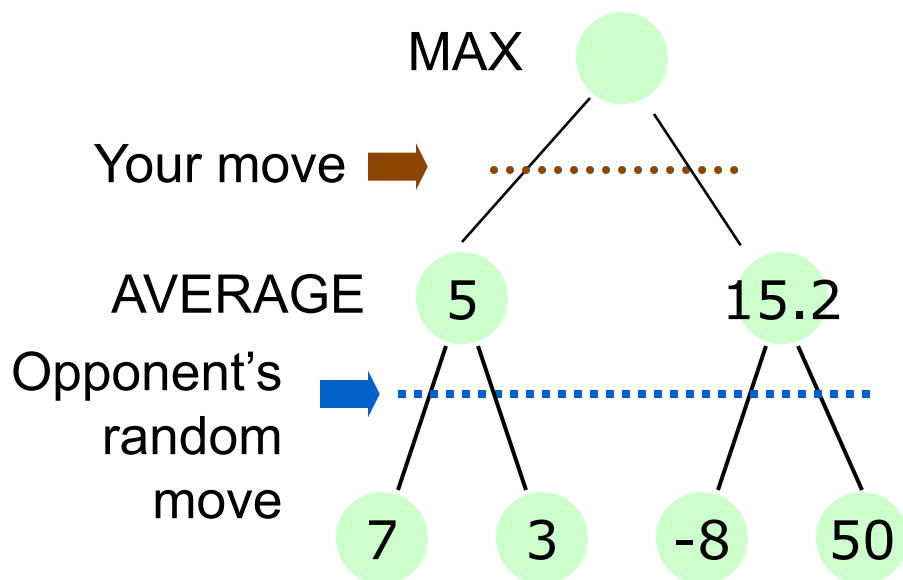
- Your opponent behaves randomly with a given probability distribution,
- If you move left, your opponent will select actions with probability [0.5,0.5]
- If you move right, your opponent will select actions with [0.6,0.4]

# Expectimax example



- Your opponent behaves randomly with a given probability distribution,
- If you move left, your opponent will select actions with probability  $[0.5, 0.5]$
- If you move right, your opponent will select actions with  $[0.6, 0.4]$

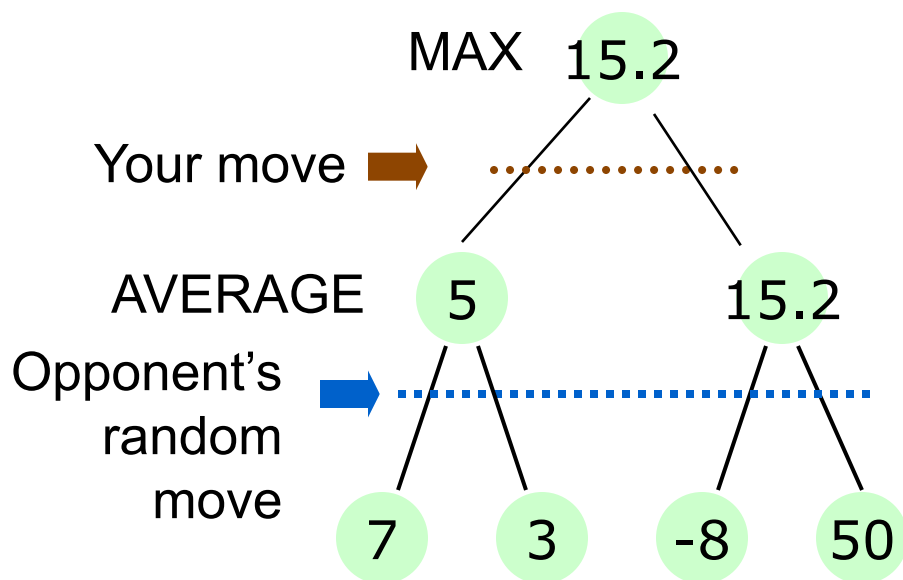
# Expectimax example



- Your opponent behaves randomly with a given probability distribution,
- If you move left, your opponent will select actions with probability  $[0.5, 0.5]$
- If you move right, your opponent will select actions with  $[0.6, 0.4]$

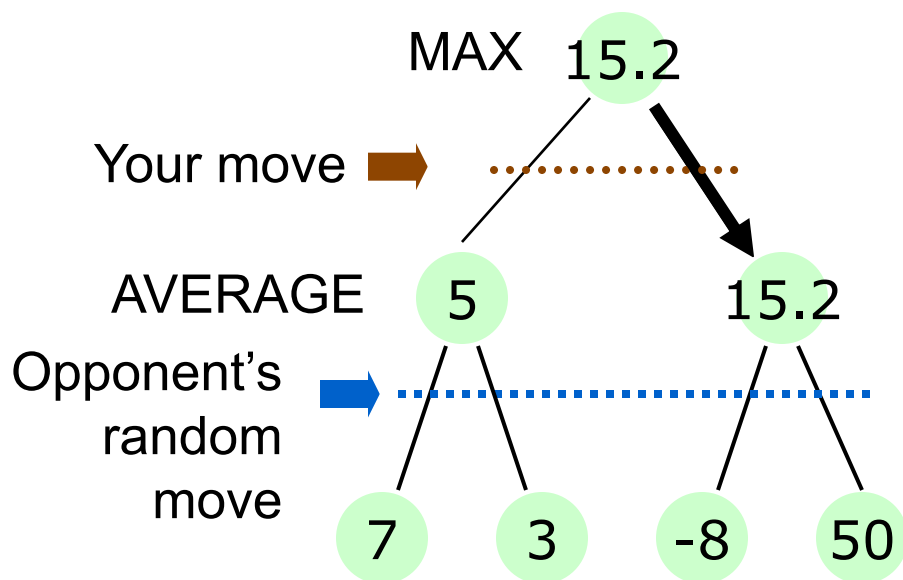


# Expectimax example



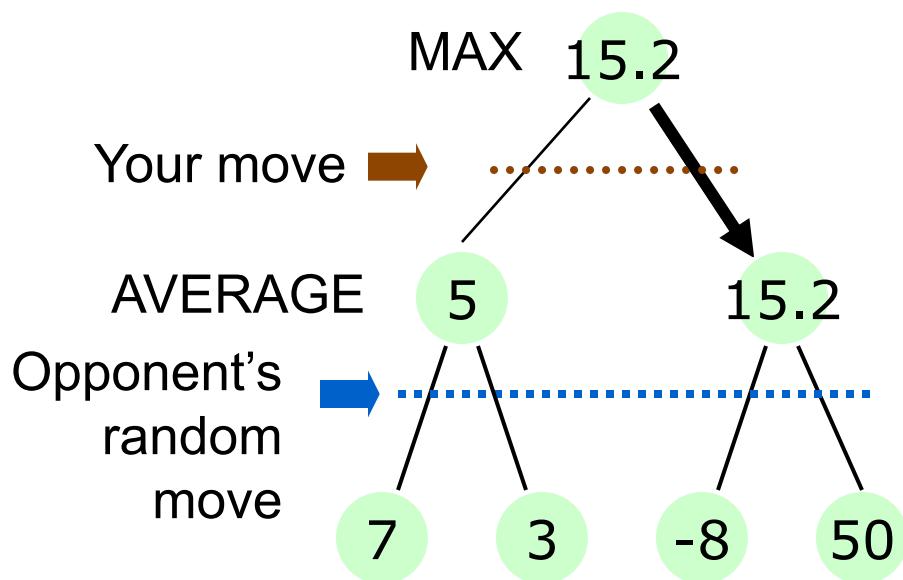
- Your opponent behaves randomly with a given probability distribution,
- If you move left, your opponent will select actions with probability [0.5,0.5]
- If you move right, your opponent will select actions with [0.6,0.4]

# Expectimax example



- Your opponent behaves randomly with a given probability distribution,
- If you move left, your opponent will select actions with probability  $[0.5, 0.5]$
- If you move right, your opponent will select actions with  $[0.6, 0.4]$

# Expectimax example



- Your opponent behaves randomly with a given probability distribution,
- If you move left, your opponent will select actions with probability  $[0.5, 0.5]$
- If you move right, your opponent will select actions with  $[0.6, 0.4]$

**Note: pruning becomes tricky in expectimax... think about why.**

# Summary of game playing

- Minimax search
- Game tree
- Alpha-beta pruning
- Early stop with an evaluation function
- Expectimax

# More reading / resources about game playing

- Required reading: AIMA 5.1-5.3
- Stochastic game / Expectiminimax: AIMA 5.5
  - Backgammon. TD-Gammon
  - Blackjack, Poker
- Famous game AI: Read AIMA Ch. 5 “Historical notes”
  - Deep blue
  - TD Gammon
  - AlphaGo / AlphaZero
- AlphaGo: <https://www.nature.com/articles/nature16961>

# About the upcoming midterm

- **Topics:** Cover up to A\*-Search and HW2. (not including Games / Adversarial search)
- **Types of questions:**
  - True/False, MCQ, BayesNet reading, simple calculations / derivations. Similar to quiz questions in the lecture slides.
- **What am I testing you on:**
  - Test of basic concepts.
  - Test of simple mechanical calculations.
  - Test of your ability to “generalize”

# About the upcoming midterm

- **Topics:** Cover up to A\*-Search and HW2. (not including Games / Adversarial search)
- **Types of questions:**
  - True/False, MCQ, BayesNet reading, simple calculations / derivations. Similar to quiz questions in the lecture slides.
- **What am I testing you on:**
  - Test of basic concepts.
  - Test of simple mechanical calculations.
  - Test of your ability to “generalize”
- **Tough questions? Don't panic and try your best.**
  - Bonus questions may not be difficult (some are), but you are expected to finish non-bonus questions first.

# About the upcoming midterm

- **Date:** Next Tuesday in class
- **Duration:** 1 hour 15 minutes
- **Format:** Open book (but no digital devices)
  - No calculator needed.



# How to prepare for the midterm?

- Read relevant chapters from AIMA
- Lecture videos from a previous year will be made available to you (on Piazza)
  - Mostly the same
  - Slight differences on the syllabus
- Try solving problems in the optional homeworks
  - HW0, HW1, HW2
- Participate in discussions on Piazza

# Time to pause and take a look back!

Week	Topic	
1	Course Overview & Intelligent Agents	
2	Machine Learning	}
	Machine Learning	
2	Machine Learning	}
	Probabilistic Graphical Models	
3	Probabilistic Graphical Models	}
	Search: Problem solving with search	
4	Search: Search algorithms	}
	Search: Minimax search and game playing	
5	Midterm Review	
	<b>Midterm</b>	
6	RL: Intro / Markov Decision Processes	}
	RL: Solving MDPs	
7	RL: Bandits and Exploration	}
	RL: Reinforcement Learning Algorithms	
8	RL: Reinforcement Learning Algorithms	}
	Logic: Propositional Logic	
9	Logic: First order Logic	}
	Responsible AI	
10	Responsible AI	}
	Final Review	
11	Final Exam	

Machine Learning

Probabilistic Reasoning

Search

Reinforcement Learning

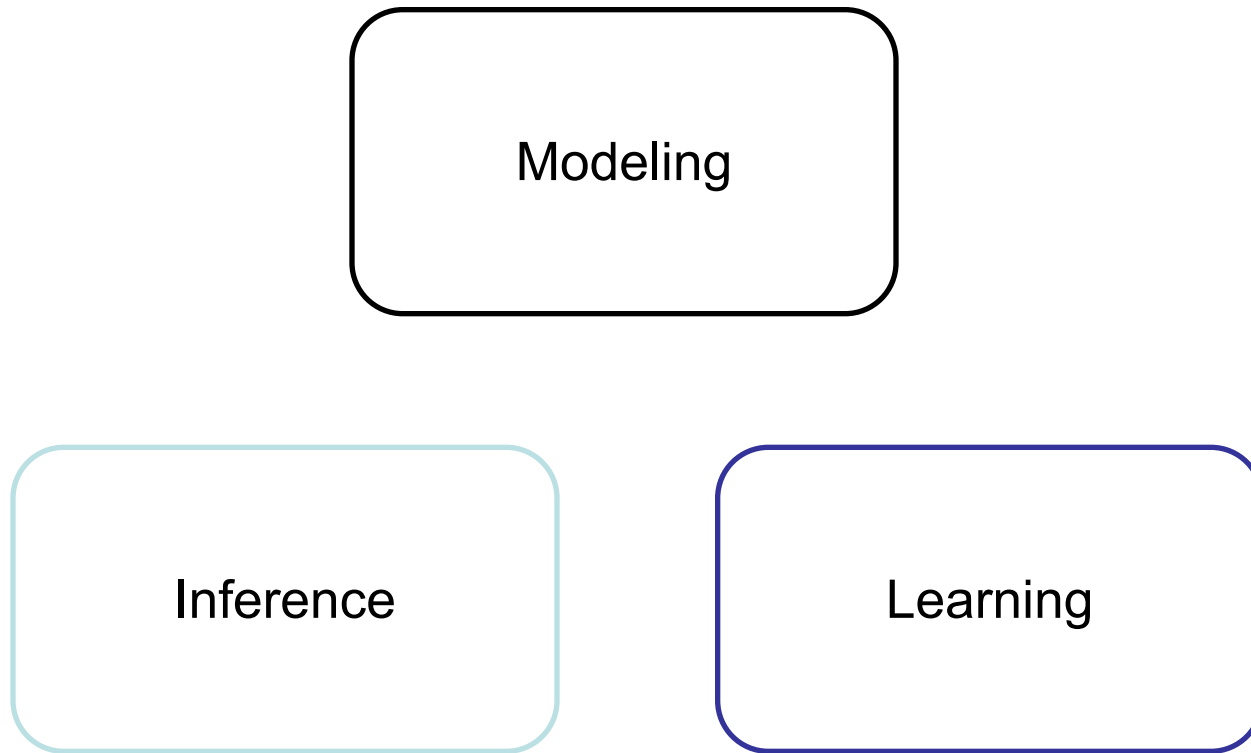
Logic

Responsible AI

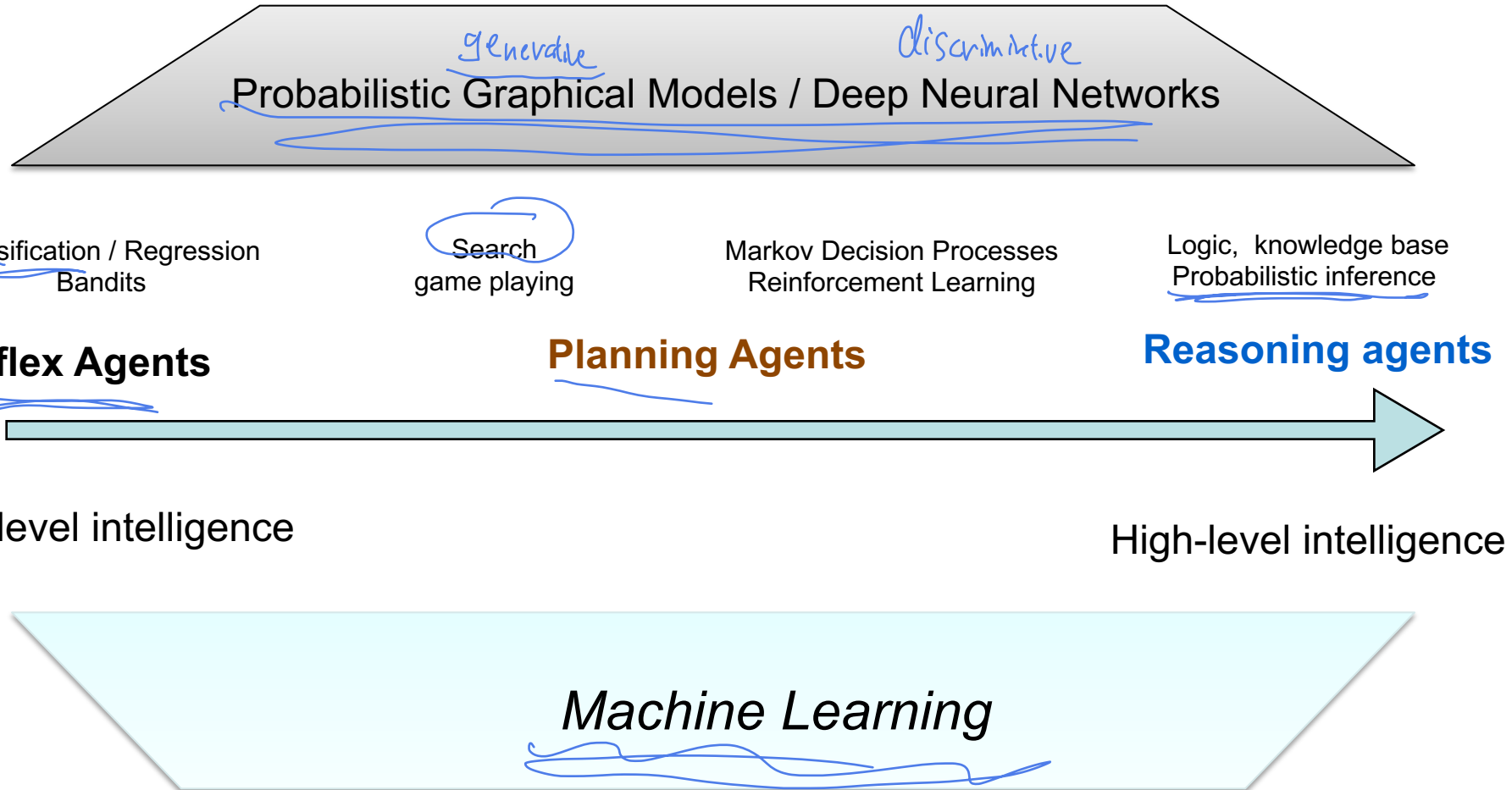
# Lecture 1: AI Overview & Agents

- AI for problem solving
- Rational agents
- Examples of AI in the real world
- Modelling-Inference-Learning Paradigm

# New paradigm: Modeling-Inference-Learning



# Structure of the course



# Modeling-Learning-Inference Paradigm

---

	<b>Modeling</b>	<b>Learning</b>	<b>Inference</b>
Classifier agent (Spam filter)	Feature engineering Hypothesis class	Minimize Error rate	Prediction on new data points
Probabilistic Inference agent (Sherlock)	Joint distribution Draw edges in BN Conditional independences	Fitting the CPTs to Data	Marginalization (conceptually easy)
Search agents	State-Space- diagram	Environment given (learn edge weights) <b>(Learn heuristics?)</b>	<b>Uninformed Search / A* Search etc...</b>

# PEAS Description of Task Environment

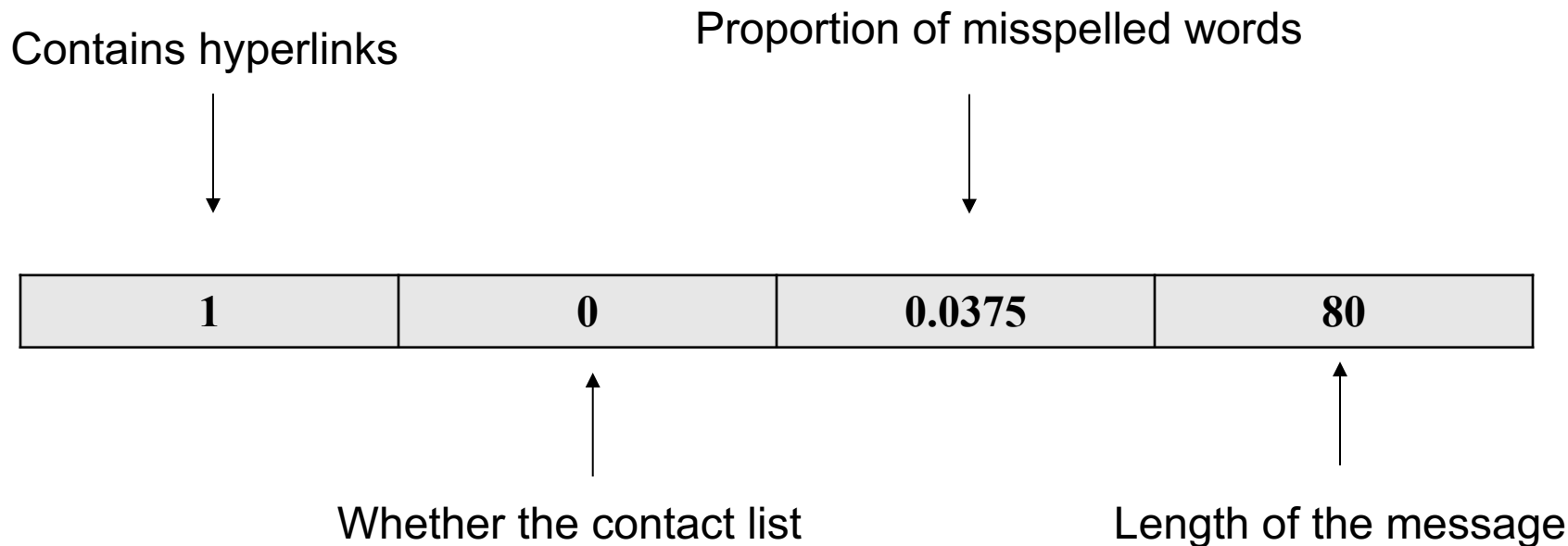
	<b>P</b>	<b>E</b>	<b>A</b>	<b>S</b>
Classifier agent (Spam filter)	Error rate	New example arrives	Classify	Observe the feature vector
Probabilistic Inference agent (Sherlock)	Error in the computed $P(Y X)$ Time/space complexity	Joint distribution via PGM	Probabilistic (Bayesian) Inference	Observe evidence
Search agents	Completeness, Optimality, Time / Space complexity	State-space diagram	Choose a sequence of operators	Observe next node

# Lecture 2-4: Machine Learning

- Machine learning
  - Types of machine learning models
  - Focus on Supervised Learning ---- classifier agents.
- What is a feature vector?
  - Feature engineering, feature extraction
- Hypothesis class and free parameters
  - How many are there? How to evaluate a classifier?
  - Error? On training data or on new data?
  - Overfitting, underfitting?
- How to learn (optimize)?
  - Surrogate losses, Gradient Descent, SGD



# Example of a feature vector of dimension 4



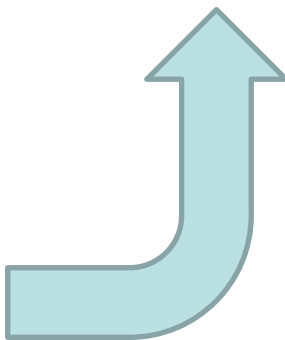
Email ADMIN January 1, 2020 at 10:35 PM EA  
([cs.ucsb.edu](http://cs.ucsb.edu)) APPLICATION -Storage Full Notes- Last -... Details

To: Yu-Xiang Wang,  
Reply-To: Email ADMIN

Dear [yuxiangw@cs.ucsb.edu](mailto:yuxiangw@cs.ucsb.edu),  
Your email has used up the storage limit of 99.9 gigabytes as defined by your Administrator. You will be blocked from sending and receiving messages if not re-validated within 48hrs.  
Kindly click on your email below for quick re-validation and additional storage will be updated automatically

[yuxiangw@cs.ucsb.edu](mailto:yuxiangw@cs.ucsb.edu)

Regards,  
E-mail Support 2020.



**Step 1 in Modelling**  
**Feature extractor:**  
Converting the object of interest to a vector of numerical values.

## Example: Linear classifiers

- $\text{Score}(x) = w_0 + w_1 * 1(\text{hyperlinks}) + w_2 * 1(\text{contact list}) + w_3 * \text{misspelling} + w_4 * \text{length}$
- A linear classifier:  $h(x) = 1$  if  $\text{Score}(x) > 0$  and 0 otherwise.
- Question: What are the “free-parameters” in a linear classifier?
  - If we redefine  $\mathcal{Y} = \{-1, 1\}$
  - A compact representation:

$$h(x) = \underline{\text{sign}(w^T [1; x])}$$

# Linear classifier

- Think geometrically

- What is the set of all feature vectors that a particular linear classifier will classify as “Spam”?

$$\{x \in \mathbb{R}^d \mid \text{sign}(w^T x) = 1\} = \{x \in \mathbb{R}^d \mid \underline{w^T x} > 0\}$$

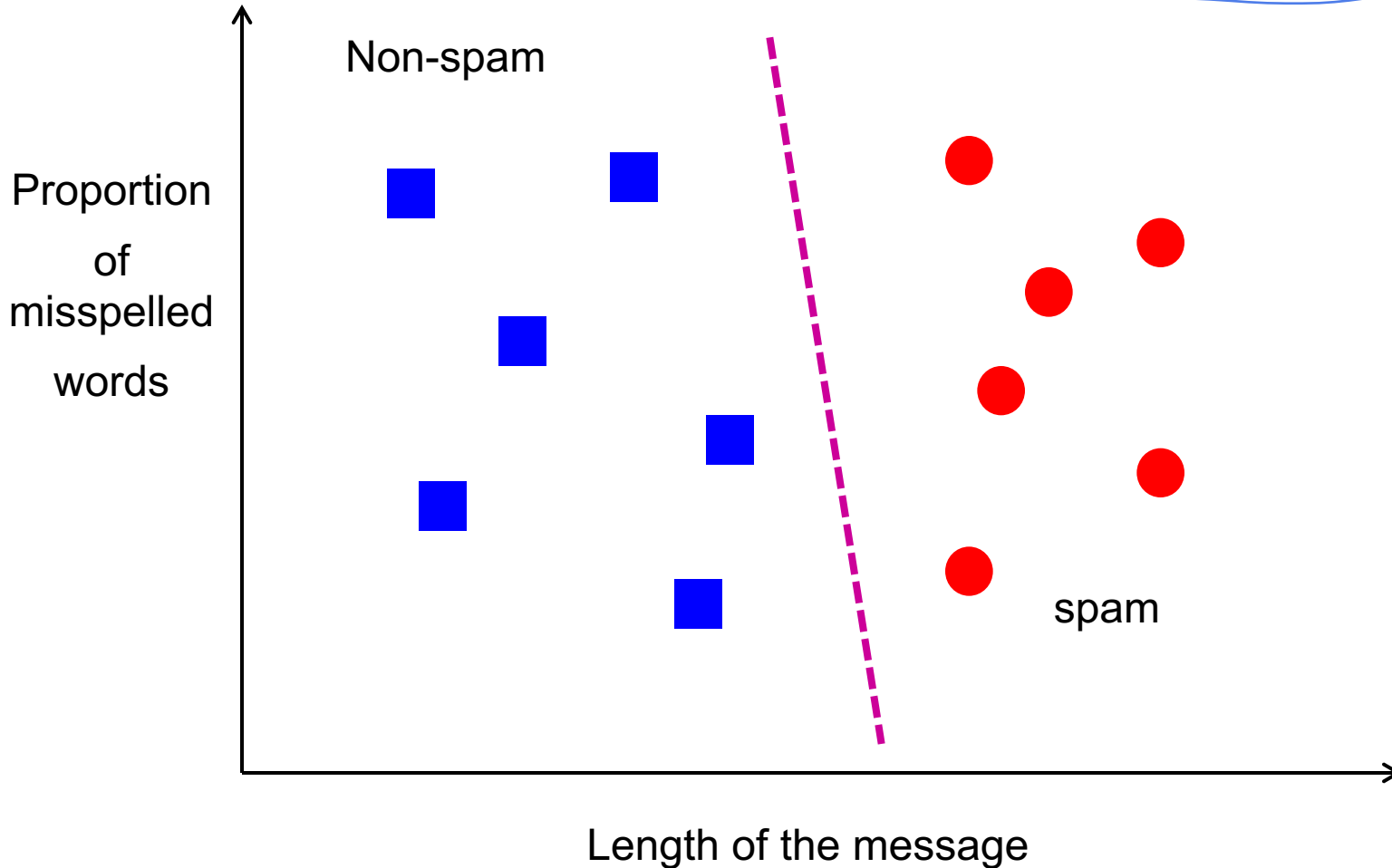
- What is the set of all classifiers that will classify a particular data point to be “Spam”?

$$\underline{W(x)} = \{w \in \mathbb{R}^d \mid \text{sign}(w^T x) = 1\} = \{w \in \mathbb{R}^d \mid w^T x > 0\}$$

# Geometric view: Linear classifier are “half-spaces”!

$$\{x \mid w_0 + w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + w_4 * x_4 > 0\}$$

The set of all “emails” that will be classified as “Spams”.



Just “relax”: relaxing a hard problem into an easier one

$$\min_{w \in \mathbb{R}^d} \text{Error}(w) = \frac{1}{n} \sum_{i=1}^n \mathbf{1}(\text{sign}(w^T x_i) \neq y_i)$$



$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \ell(w^T x_i, y_i).$$

# Loss functions and surrogate losses

# Loss functions and surrogate losses

- 0-1 loss:  $\mathbf{1}(h_w(x) \neq y)$

# Loss functions and surrogate losses

- 0-1 loss:  $\mathbf{1}(h_w(x) \neq y) = \mathbf{1}(\text{sign}(S_w(x)) \neq y)$



# Loss functions and surrogate losses

- 0-1 loss:  $\mathbf{1}(h_w(x) \neq y) = \mathbf{1}(\text{sign}(S_w(x)) \neq y)$
- Square loss:  $(y - S_w(x))^2$

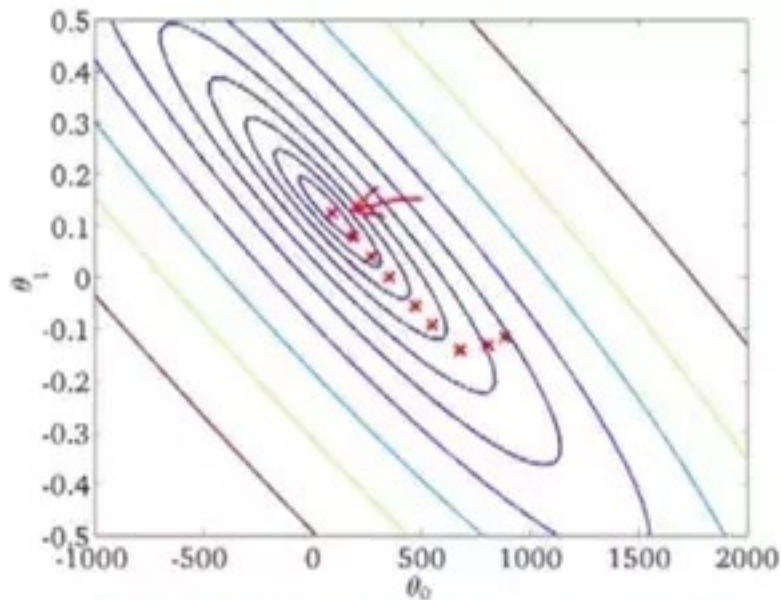
# Loss functions and surrogate losses

- 0-1 loss:  $\mathbf{1}(h_w(x) \neq y) = \mathbf{1}(\text{sign}(S_w(x)) \neq y)$
- Square loss:  $(y - S_w(x))^2$
- Logistic loss:  $\log_2(1 + \exp(-y \cdot S_w(x)))$

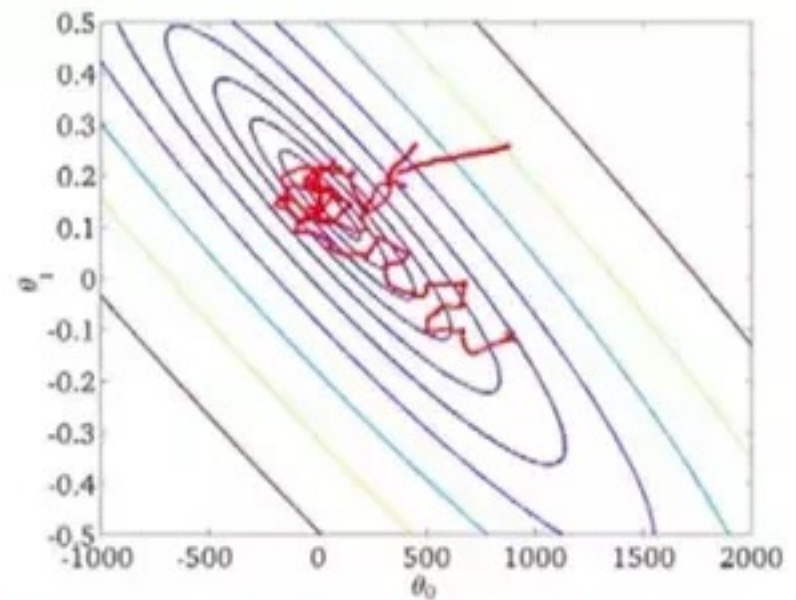
# Loss functions and surrogate losses

- 0-1 loss:  $\mathbf{1}(h_w(x) \neq y) = \mathbf{1}(\text{sign}(S_w(x)) \neq y)$
- Square loss:  $(y - S_w(x))^2$
- Logistic loss:  $\log_2(1 + \exp(-y \cdot S_w(x)))$
- Hinge loss:  $\max(0, 1 - y \cdot S_w(x))$

# Illustration of GD vs SGD



**Batch Gradient Descent**



**Stochastic Gradient Descent**

**Observation:** With the time gradient descent taking one step.

SGD would have already moved many steps.

# One natural stochastic gradient to consider in machine learning

- Recall that

$$\min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \ell(\theta, (x_i, y_i))$$

- Pick a **single** data point i uniformly at random

- Use  $\nabla_{\theta} \ell(\theta, (x_i, y_i))$

- Show that this is an unbiased estimator!

# Intuition of the SGD algorithm on the “Spam Filter” example

$$\nabla \ell(w, (x_i, y_i)) = \underbrace{\frac{\exp(-y_i \cdot x_i^T w)}{1 + \exp(-y_i \cdot x_i^T w)}}_{\text{Scalar } > 0:} \cdot \underbrace{(-y_i x_i)}_{\text{Vector of dimension } d:}$$

Scalar > 0:  
≈ 0 if the prediction is correct  
≈ 1 otherwise

Vector of dimension d:  
provides the direction of the gradient

If we receive an example [1, 0 , 0.0375, 80] like the one before.  
And a label  $y = 1$  saying that this is a spam.

**How will the SGD update change the weight vector?**

Then by moving  $w$  towards the negative gradient direction, we are changing the weight vector by increasing the weights. i.e., increasing the amount they contribute to the score function (if currently the classifier is making a mistake on this example)

# Calculating the gradients of surrogate loss functions

$$y \in \{-1, 1\}$$

$$l(w) = \log(1 + \exp(-y \cdot x^T w))$$

- Example: Binary logistic loss

$$\nabla l(w) = \frac{\exp(-y \cdot x^T w)}{1 + \exp(-y \cdot x^T w)} \cdot (-y) \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix}$$

- Example: Cross-entropy loss

$$y \in \{0, 1\}$$

$$l(w) = - \left( y \cdot \log \frac{\exp(x^T w)}{1 + \exp(x^T w)} + (1-y) \cdot \log \frac{1}{1 + \exp(x^T w)} \right)$$

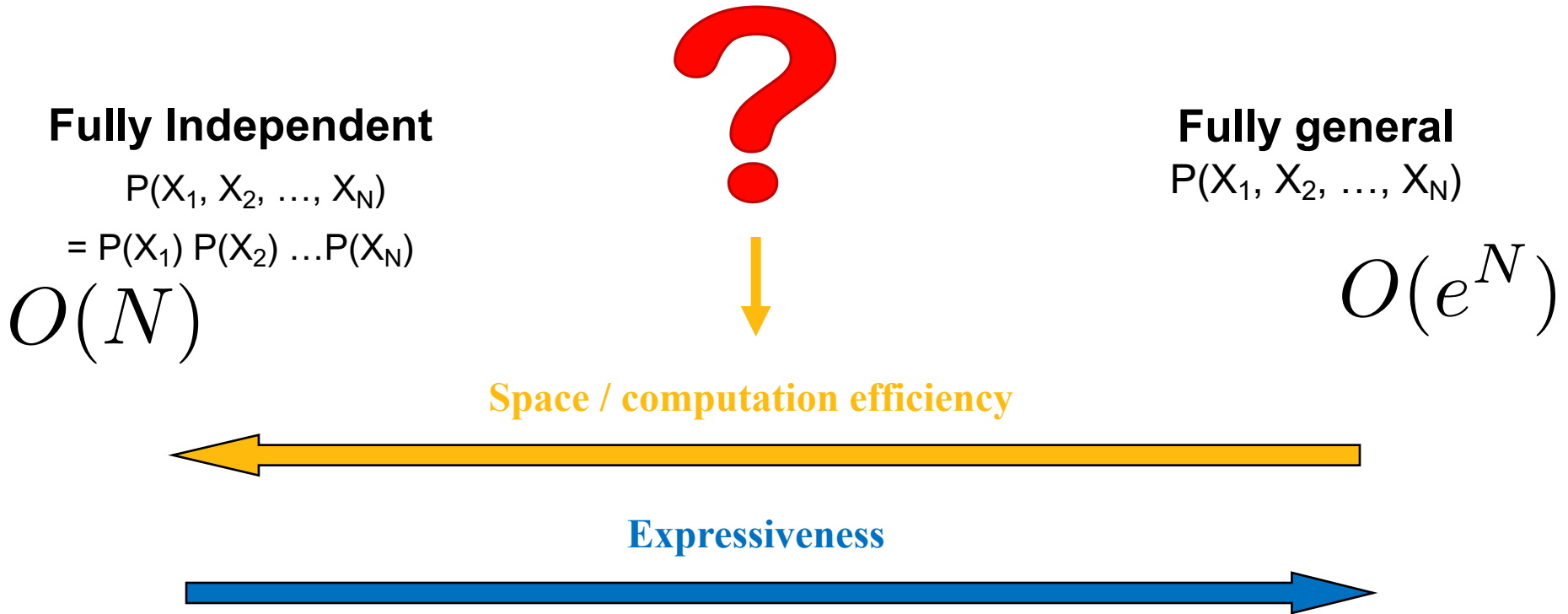
$$\begin{aligned} \nabla l(w) &= -y \cdot \frac{\exp(x^T w)}{(1 + \exp(x^T w))^2} \cdot \exp(-x^T w) \cdot (-\vec{x}) + (1-y) \cdot \frac{-\exp(x^T w)}{(1 + \exp(x^T w))^2} \cdot \vec{x} \\ &= - \left( y \cdot \frac{1}{1 + \exp(x^T w)} \cdot (-\vec{x}) + (1-y) \cdot \frac{\exp(x^T w)}{1 + \exp(x^T w)} \cdot \vec{x} \right) \end{aligned}$$

# Lecture 5-7: Probabilities and BayesNet

- Modelling the world with a joint probability distribution
  - Number of independent parameters?
- CPTs
  - Count number of independent numbers to represent a CPT
- Conditional, Marginal, Probabilistic Inference with Bayes Rule
- Read off conditional independences from the graph
  - d-separation
  - Bayes ball algorithm
  - Markov Blanket



# Tradeoffs in our model choices



## Idea:

1. Independent groups of variables?
2. Conditional independences?

# How should we connect the nodes? (3 min discussion)

Burglary

Earthquake

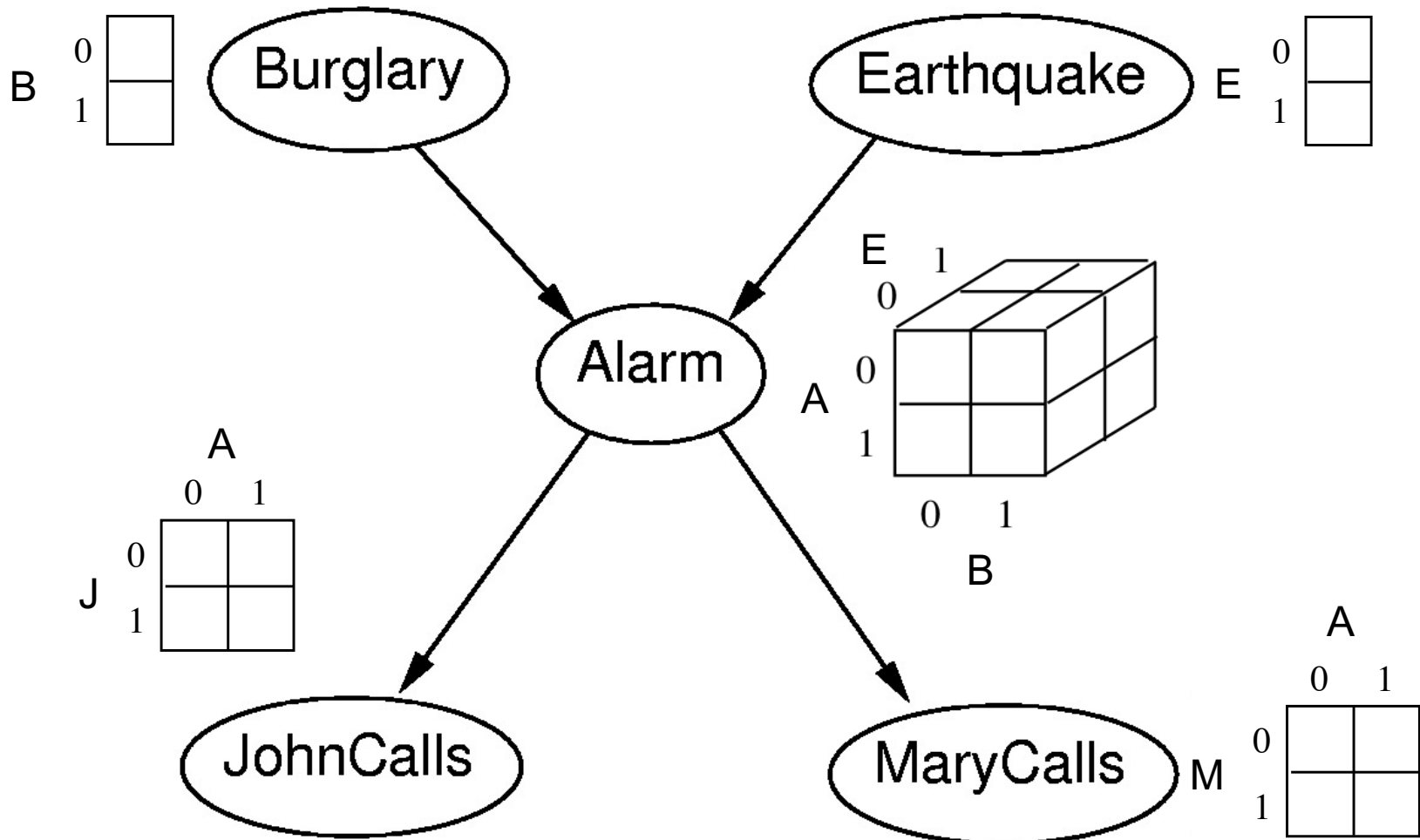
Alarm

JohnCalls

MaryCalls

Links and CPTs?

# What are the CPTs? What are their dimensions?

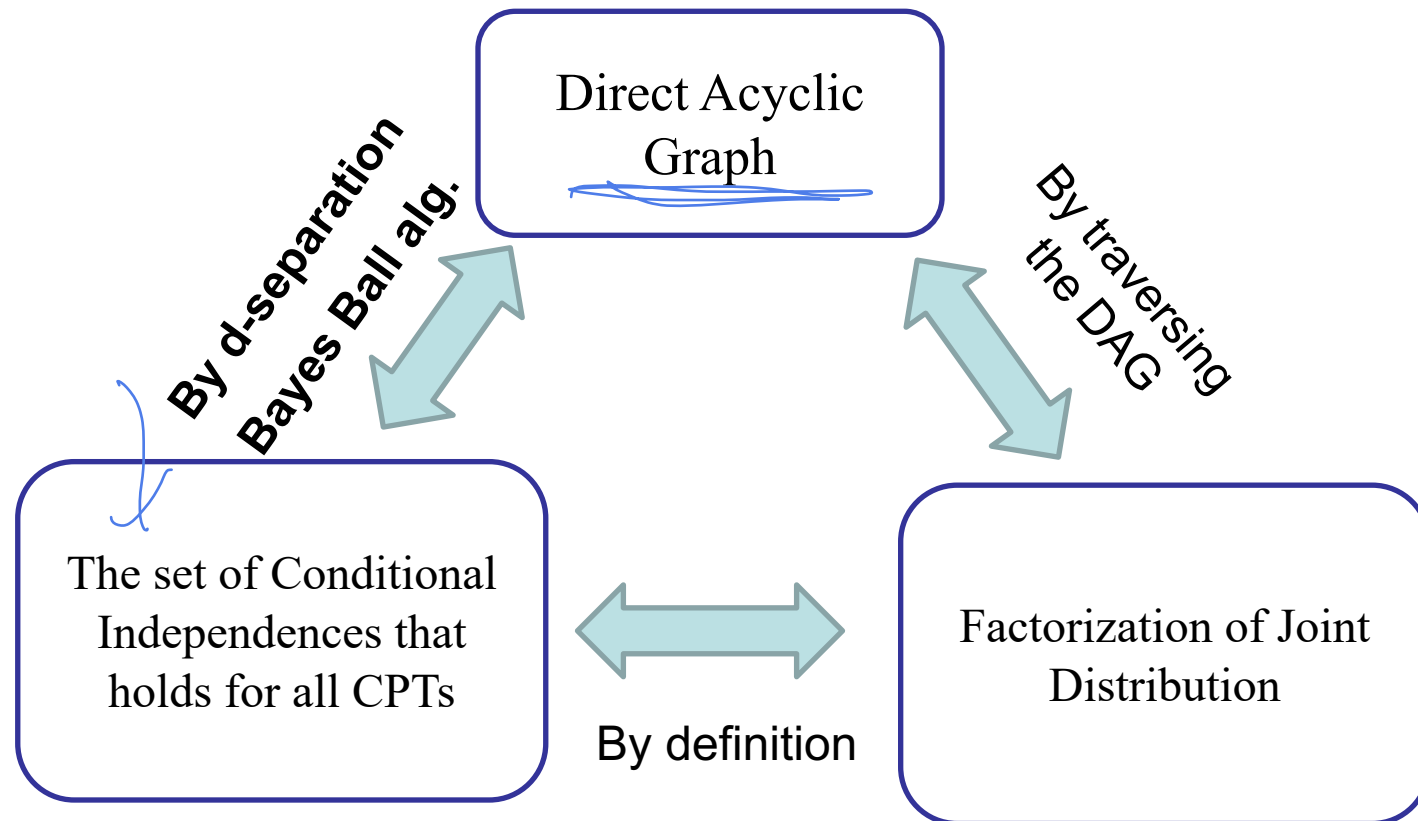


**Question: How to fill values into these CPTs?**

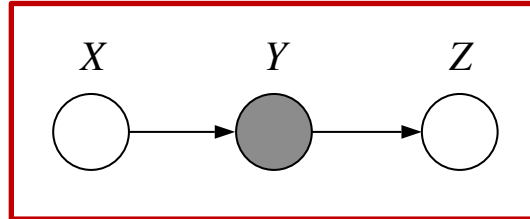
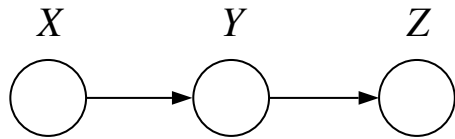
**Ans: Specify by hands. Learn from data (e.g., MLE).**

# Big question: Is there a general way that we can answer questions about conditional independences by just inspecting the graphs?

- Turns out the answer is “Yes!”

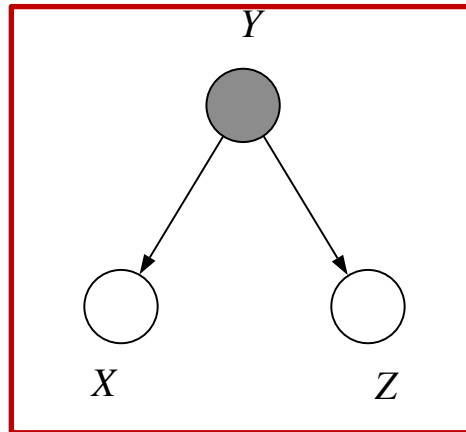
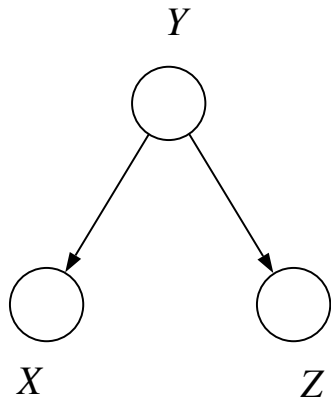


# d-separation in three canonical graphs



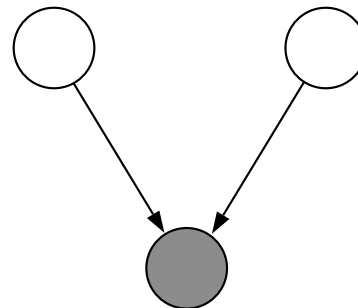
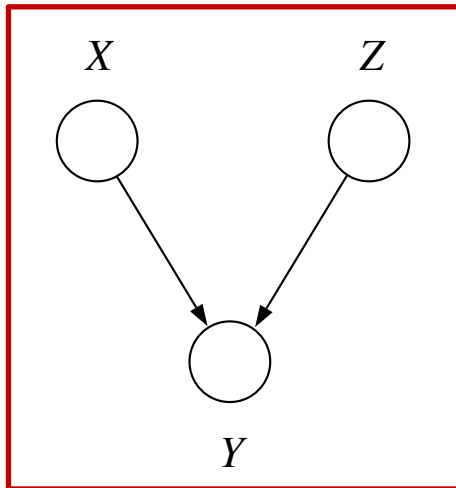
$$X \perp Z | Y$$

“X and Z are d-separated by the observation of Y.”



$$X \perp Z | Y$$

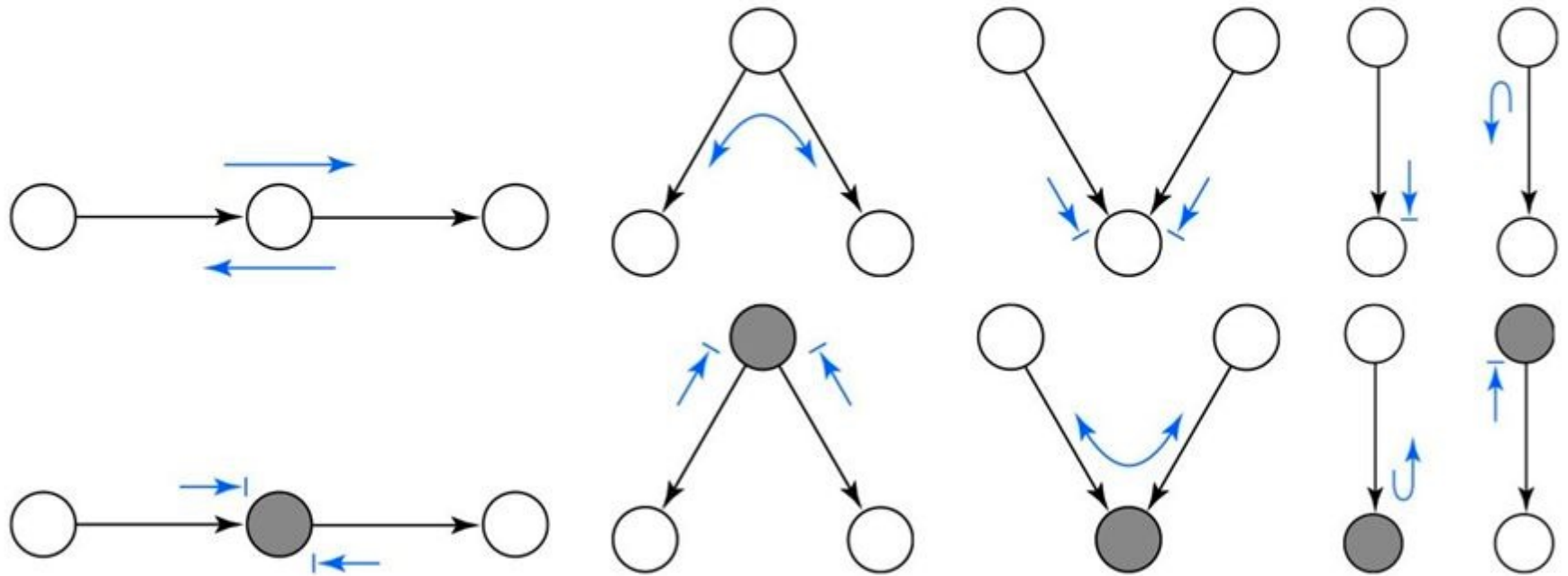
“X and Z are d-separated by the observation of Y.”



$$X \perp Z | Y$$

“X and Z are d-separated by NOT observing Y nor any descendants of Y.”

# The Ten Rules of Bayes Ball Algorithm



# Lecture 7-9: Search

- Problem solving by search
  - Abstraction, problem formulation
  - State-space diagram
  - Count the number of states, number of actions.
- Search algorithms
  - General search strategy, data-structure used
  - Space / time complexity
  - Four evaluation criteria
- Heuristic search
  - When does it work?

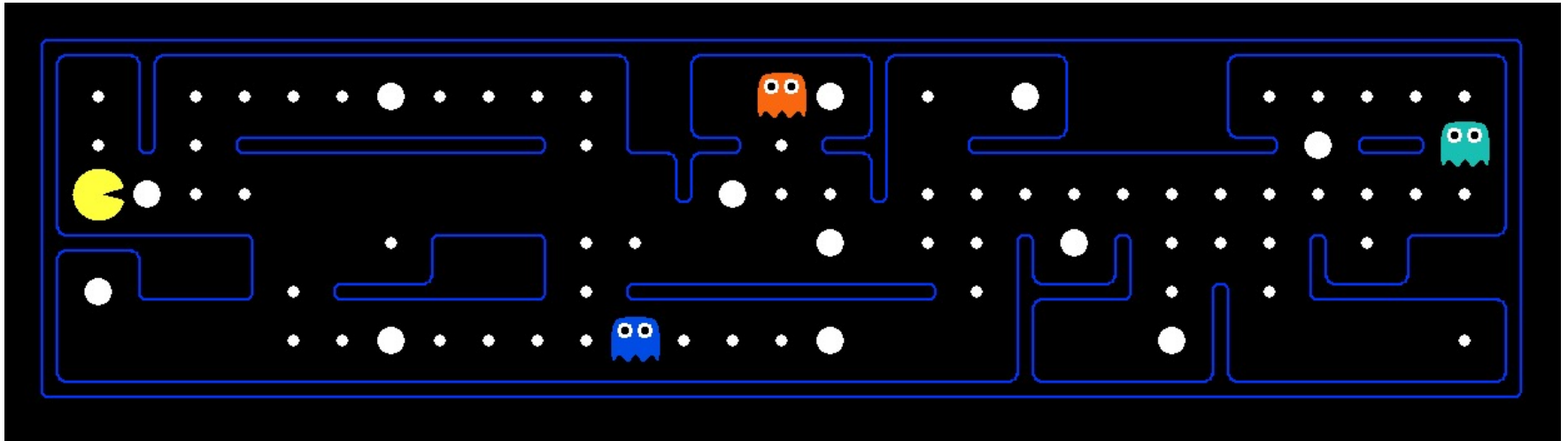
# Problem Formulation and Search

- Problem formulation
  - State-space description  $\langle \{S\}, S_0, \{S_G\}, \{O\}, \{g\} \rangle$ 
    - **S**: Possible states
    - **S<sub>0</sub>**: Initial state of the agent
    - **S<sub>G</sub>**: Goal state(s)
      - Or equivalently, a goal test **G(S)**
    - **O**: Operators  $O: \{S\} \Rightarrow \{S\}$ 
      - Describes the possible actions of the agent
    - **g**: Path cost function, assigns a cost to a path/action
- At any given time, which possible action **O<sub>i</sub>** is best?
  - Depends on the goal, the path cost function, the future sequence of actions....
- Agent's strategy: Formulate, Search, and Execute
  - This is *offline* problem solving



## More quizzes: PACMAN with static ghosts

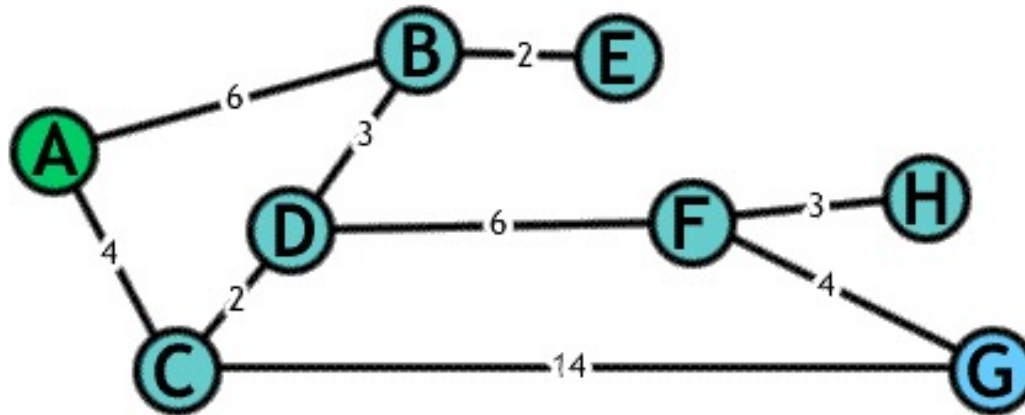
- The goal is to eat all pellets as quickly as possible while staying alive. Eating the “Power pellet” will allow the pacman to eat the ghost.



- Think about how to formulate this problem.

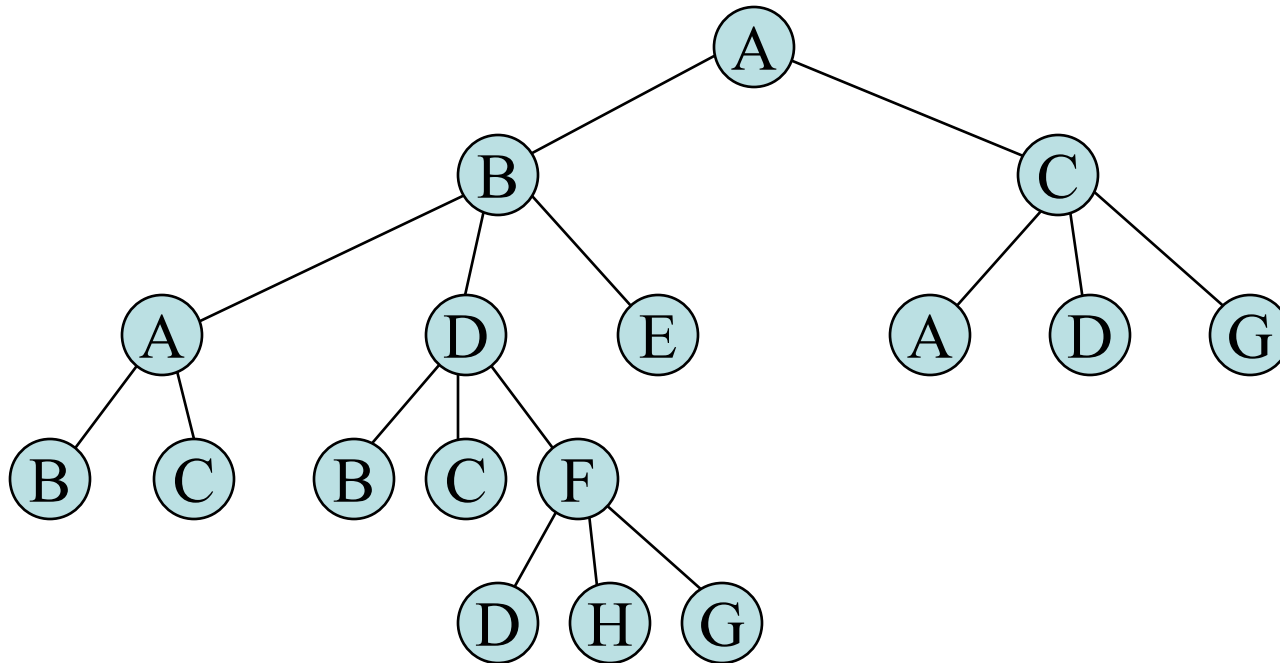
# State-Space Diagrams

- State-space description can be represented by a state-space diagram, which shows
  - States (incl. initial and goal)
  - Operators/actions (state transitions)
  - Path costs



# State Space vs. Search Tree (cont.)

Search tree (partially expanded)



# General Search Algorithm

- Uses a queue (a list) and a **queuing function** to implement a *search strategy*
  - **Queuing-Fn**(*queue*, *elements*) inserts a set of elements into the queue and determines the order of node expansion

**function GENERAL-SEARCH**(*problem*, **QUEUING-FN**) **returns** a solution or failure

*nodes* ← MAKE-QUEUE(MAKE-NODE(INITIAL-STATE[*problem*]))

**loop do**

**if** *nodes* is empty **then return** failure

*node* ← REMOVE-FRONT(*nodes*)

**if** GOAL-TEST[*problem*] applied to STATE(*node*) succeeds **then return** *node*

*nodes* ← **QUEUING-FN**(*nodes*, EXPAND(*node*, OPERATORS[*problem*]))

**end**

# Summary table of uninformed search

Criteria	BFS	Uniform-cost	DFS	Depth-limited	IDS	Bidirectional
Complete?	Yes <sup>#</sup>	Yes <sup>#&amp;</sup>	No	No	Yes <sup>#</sup>	Yes <sup>#+</sup>
Time	$O(b^d)$	$O(b^{1+[C^*/e]})$	$O(b^m)$	$O(b^l)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+[C^*/e]})$	$O(bm)$	$O(bl)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes <sup>\$</sup>	Yes	No	No	Yes <sup>\$</sup>	Yes <sup>\$+</sup>

$b$ : Branching factor

$d$ : Depth of the shallowest goal

$l$ : Depth limit

$m$ : Maximum depth of search tree

$e$ : The lower bound of the step cost

<sup>#</sup>: Complete if  $b$  is finite

<sup>&</sup>: Complete if step cost  $\geq e$

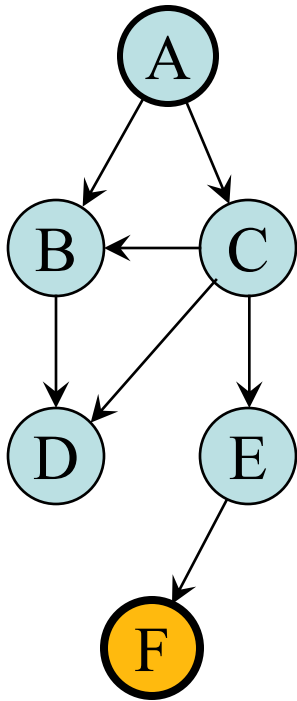
<sup>\$</sup>: Optimal if all step costs are identical

<sup>+</sup>: If both direction use BFS

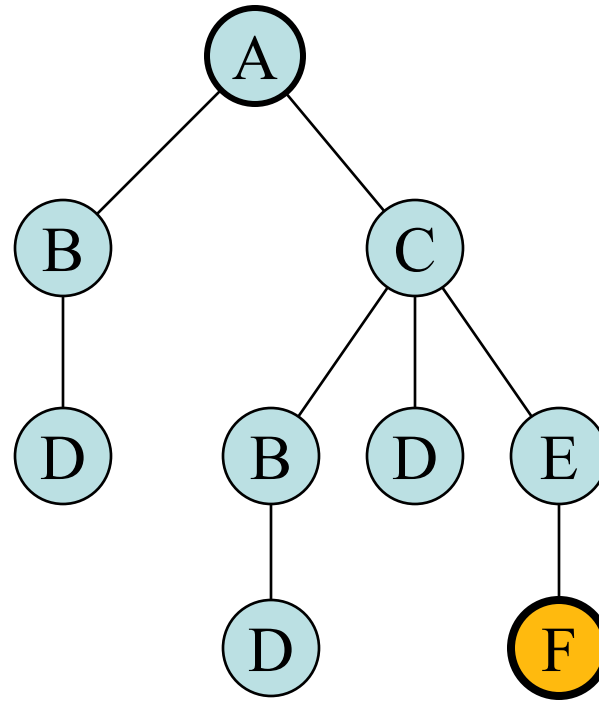
(Section 3.4.6 in the AIMA book.)

# Example

State space graph

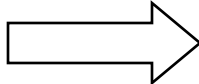


Search tree

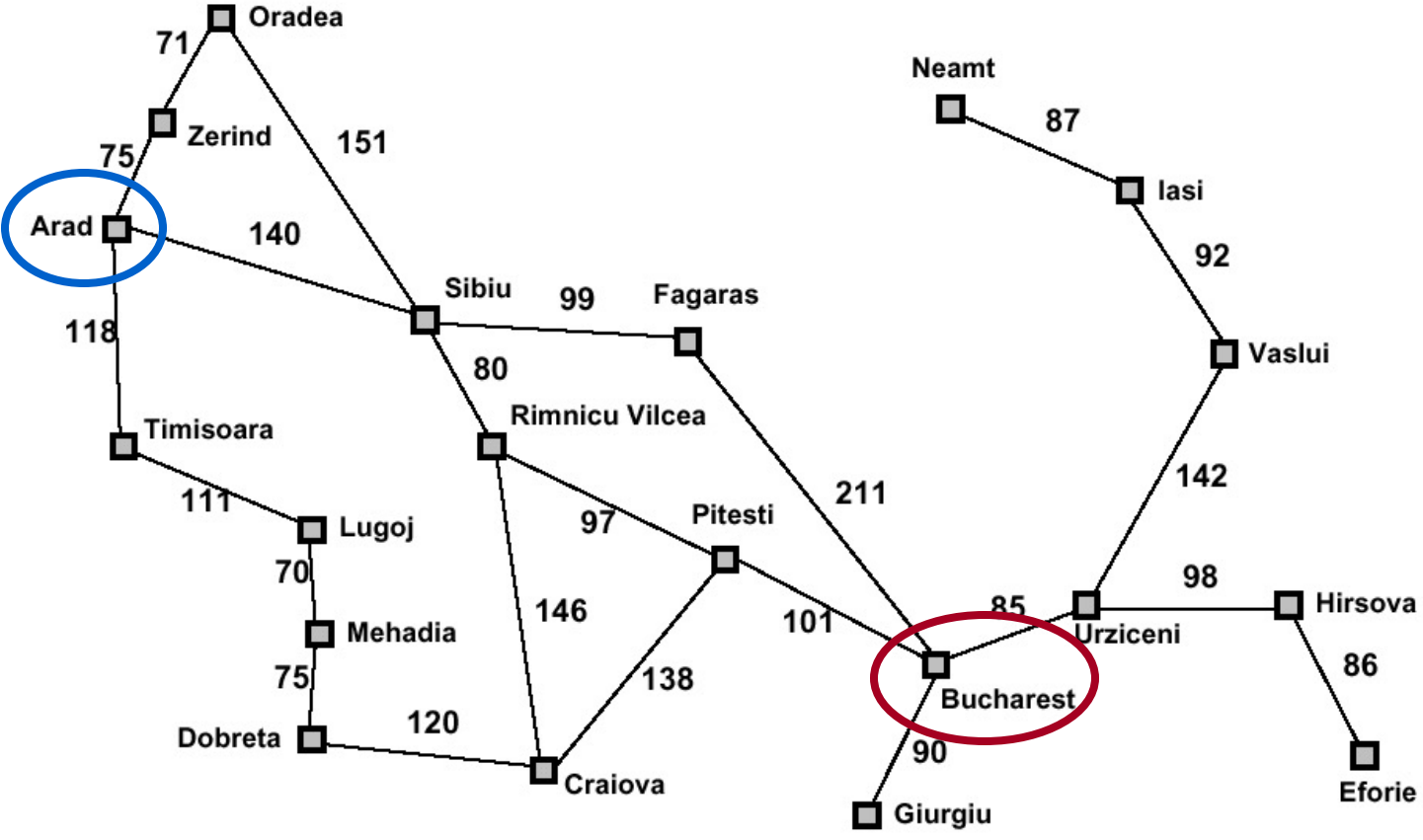


Queue

- (A)
- (B C)
- (C D)
- (D B D E)
- (B D E)
- (D E D)
- (E D)
- (D F)
- (F)
- ( )



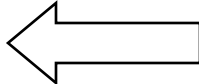
# A\* Example



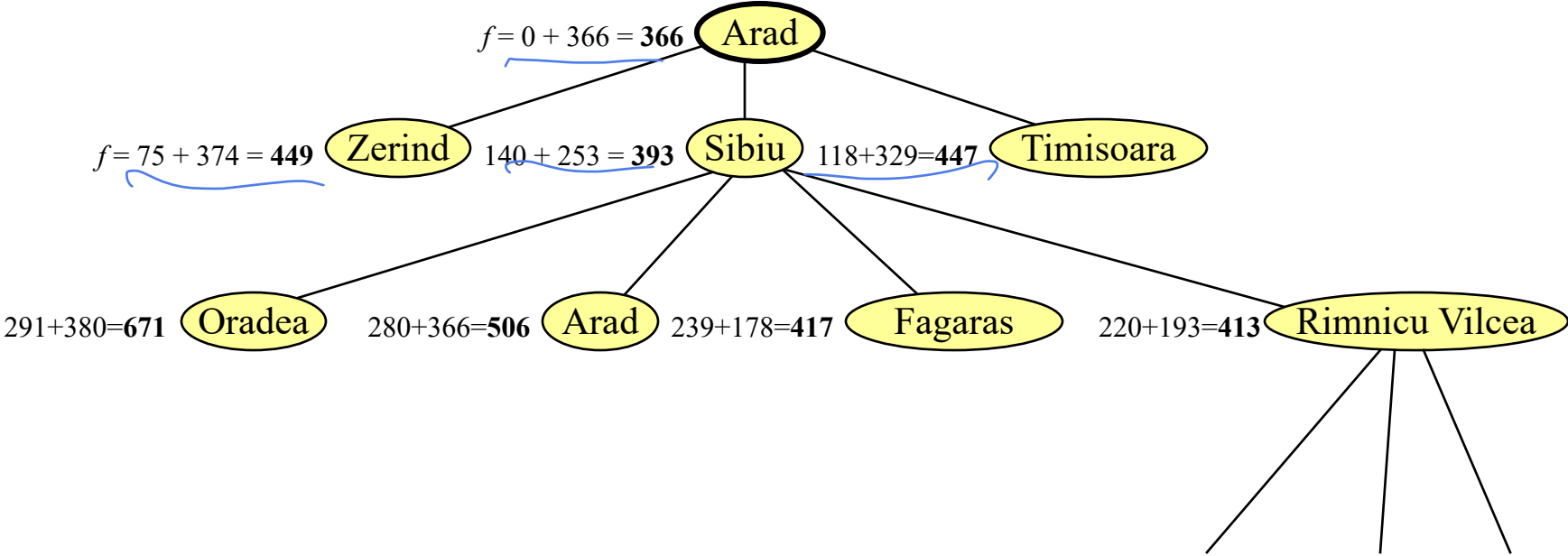
Straight-line distance to Bucharest

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

$$f(n) = g(n) + h(n)$$



# A\* Example



1. Know how to track the nodes in the frontier
2. Select which one to expand first.



# What are some general principles behind AI?

Week	Topic	
1	Course Overview & Intelligent Agents	
2	Machine Learning	Machine Learning
2	Machine Learning	
2	Machine Learning	Probabilistic Reasoning
2	Probabilistic Graphical Models	
3	Probabilistic Graphical Models	Search
3	Search: Problem solving with search	
4	Search: Search algorithms	
4	Search: Minimax search and game playing	
5	Midterm Review	
5	<b>Midterm</b>	
6	RL: Intro / Markov Decision Processes	Reinforcement Learning
6	RL: Solving MDPs	
7	RL: Bandits and Exploration	
7	RL: Reinforcement Learning Algorithms	
8	RL: Reinforcement Learning Algorithms	Logic
8	Logic: Propositional Logic	
9	Logic: First order Logic	Responsible AI
9	Responsible AI	
10	Responsible AI	
10	Final Review	
11	Final Exam	

# Optimization perspective of AI

# Optimization perspective of AI

- A rational agent  $\max_{a_1, \dots, a_T} \text{Utility}(a_1, \dots, a_T)$ 
  - **Modelling tools:** Features / Hypothesis class, PGM, State-space abstraction, agent categorization
  - **Constraints:** Computation, Data, Storage

# Optimization perspective of AI

- A rational agent  $\max_{a_1, \dots, a_T} \text{Utility}(a_1, \dots, a_T)$ 
  - **Modelling tools:** Features / Hypothesis class, PGM, State-space abstraction, agent categorization
  - **Constraints:** Computation, Data, Storage
- Discrete optimization  $\min_{p \in \text{Paths}} \text{Distance}(p)$ 
  - **Algorithmic tool:** Search / Dynamic programming

# Optimization perspective of AI

- A rational agent  $\max_{a_1, \dots, a_T} \text{Utility}(a_1, \dots, a_T)$ 
  - **Modelling tools:** Features / Hypothesis class, PGM, State-space abstraction, agent categorization
  - **Constraints:** Computation, Data, Storage
- Discrete optimization  $\min_{p \in \text{Paths}} \text{Distance}(p)$ 
  - **Algorithmic tool:** Search / Dynamic programming
- Continuous optimization  $\min_{\theta \in \mathbb{R}^d} \text{TrainingError}(\theta)$ 
  - **Algorithmic tool:** Gradient descent / Stochastic gradient descent

# Different objectives to optimize in AI

- PGM:

- MLE: Maximize the log-likelihood function
- Classifier / decision: max the posterior distribution

$$\underset{\text{spam}}{\text{arg max}} P(\text{spam} \mid \text{email text})$$

- Search and planning:

- Find valid solutions with smallest path cost.

- Machine Learning

- (Regularized) Empirical Risk Minimization (ERM):

$$\min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \ell(\theta, (x_i, y_i)) + \lambda r(\theta)$$

A lot of these problems are computationally / statistically hard, but so what?

# A lot of these problems are computationally / statistically hard, but so what?

- Get help from human:
  - Use model
  - Use abstractions at the right level
  - Use features
  - Use heuristic functions



# A lot of these problems are computationally / statistically hard, but so what?

- Get help from human:
  - Use model
  - Use abstractions at the right level
  - Use features
  - Use heuristic functions
- Get help from mathematics and computer science theory:
  - Solve an approx. solution
  - Approximate inference of a PGM
  - More iterations with less accuracy per SGD
  - A\* Search

Good luck with your midterm!