Artificial Intelligence

CS 165A

Nov 3, 2020

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

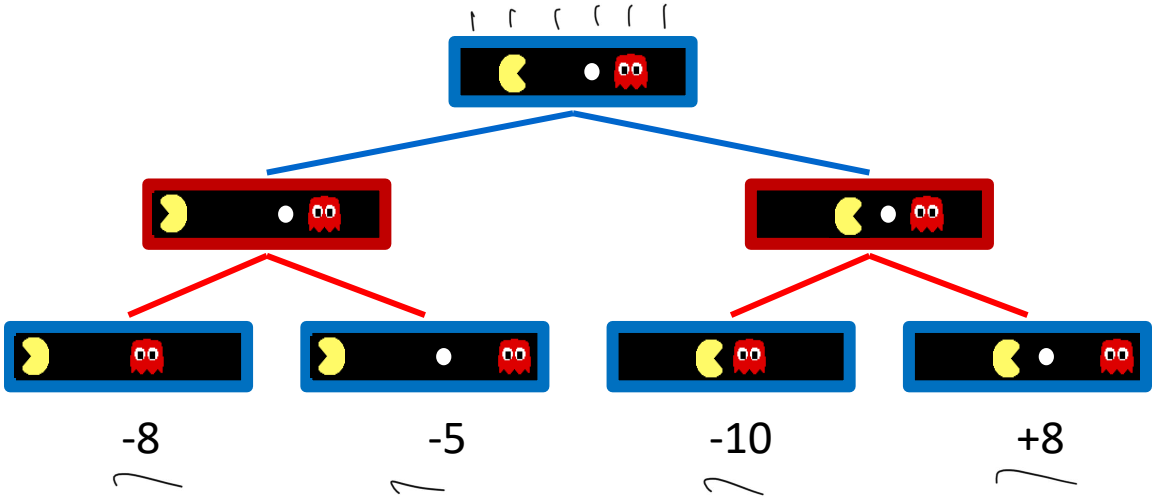
Small Pacman Example

MAX nodes: under Agent's control

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

MIN nodes: under Opponent's control

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



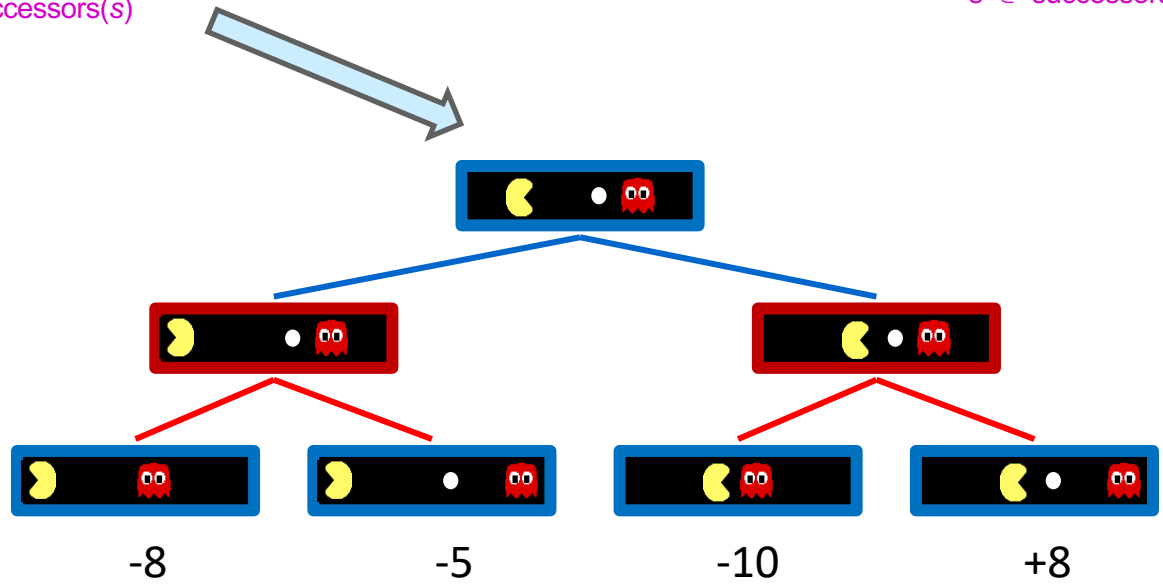
Small Pacman Example

MAX nodes: under Agent's control

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

MIN nodes: under Opponent's control

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



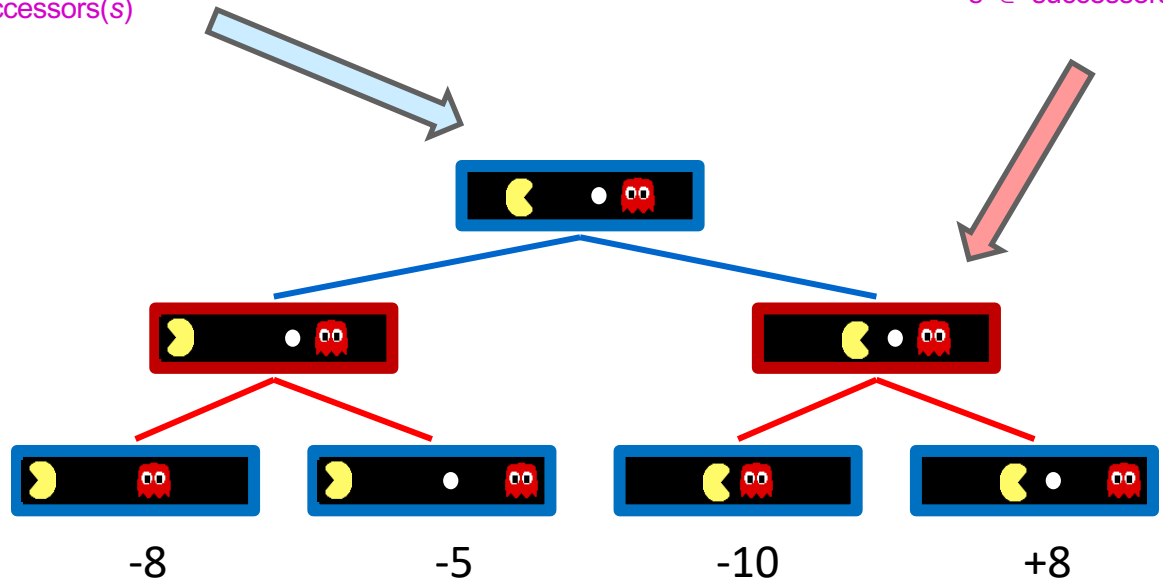
Small Pacman Example

MAX nodes: under Agent's control

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

MIN nodes: under Opponent's control

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



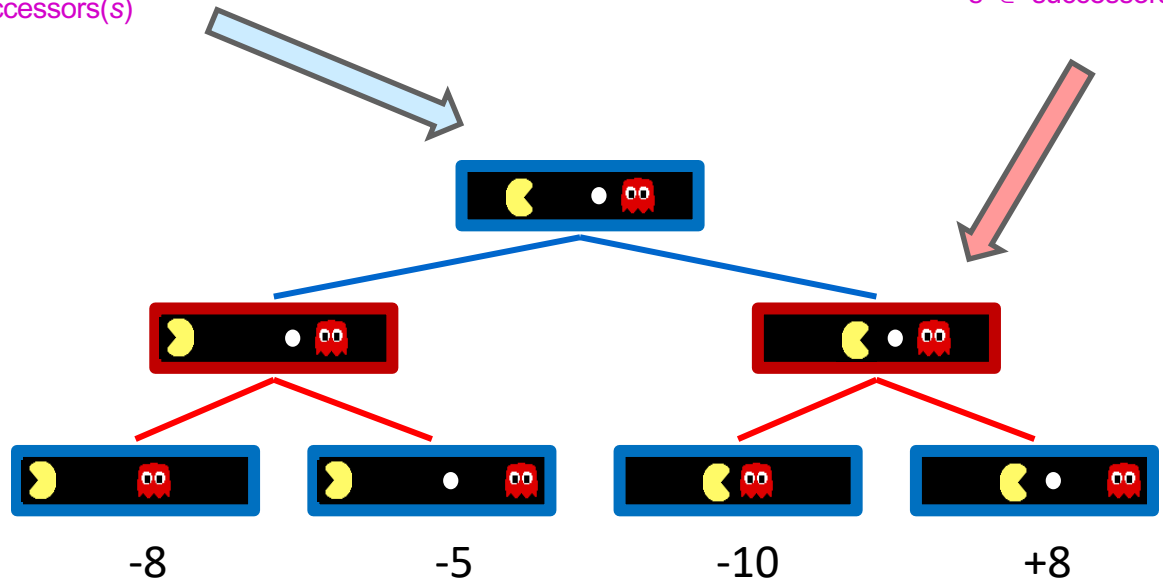
Small Pacman Example

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:

$V(s)$ = known

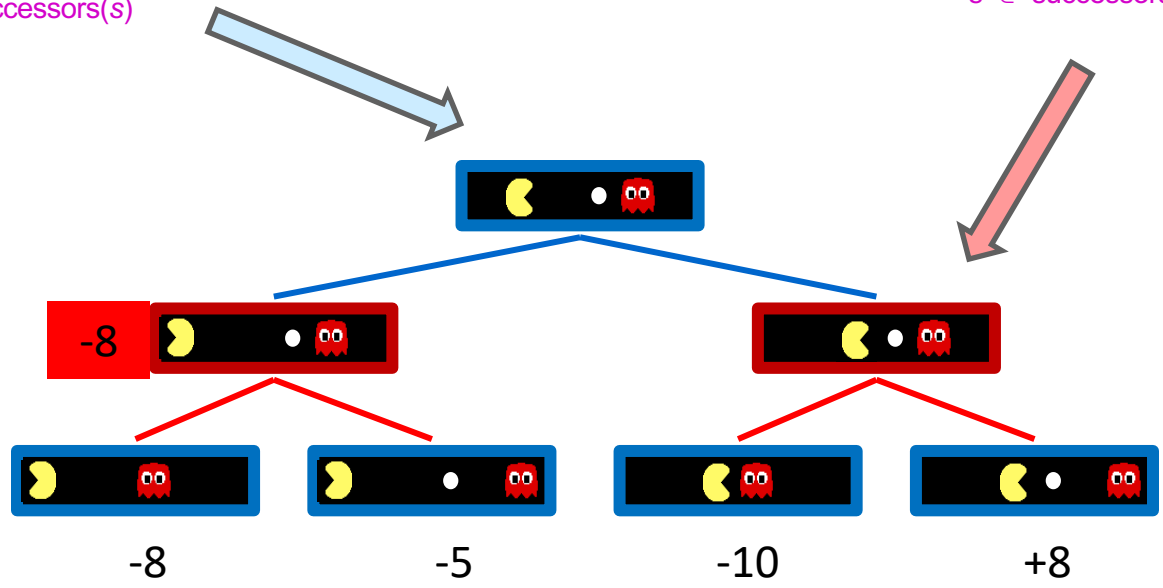
Small Pacman Example

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:

$V(s)$ = known

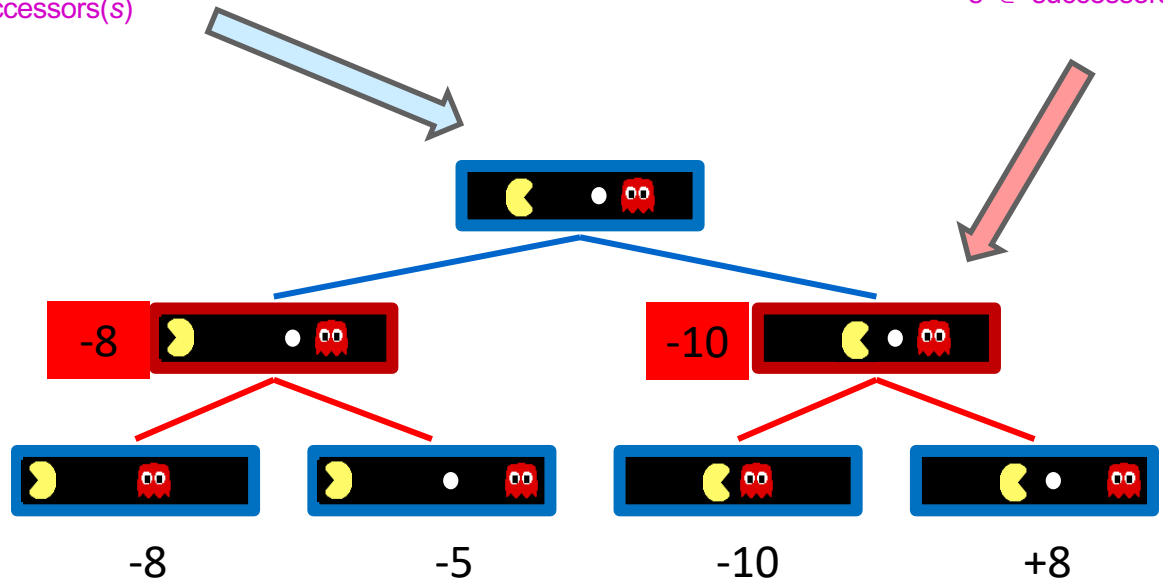
Small Pacman Example

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:

$V(s)$ = known

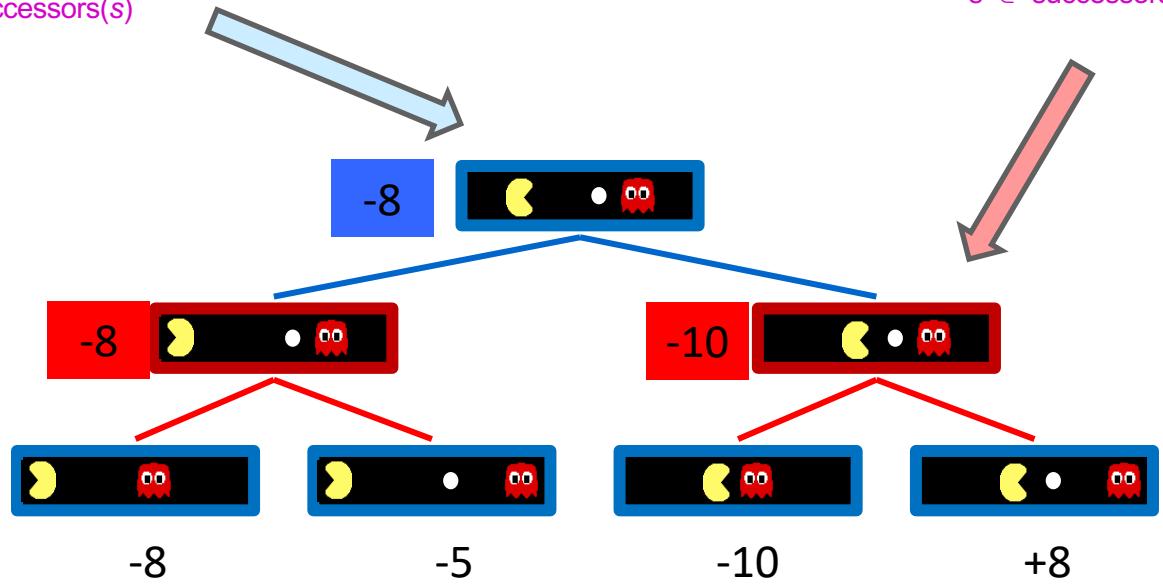
Small Pacman Example

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:

$V(s)$ = known

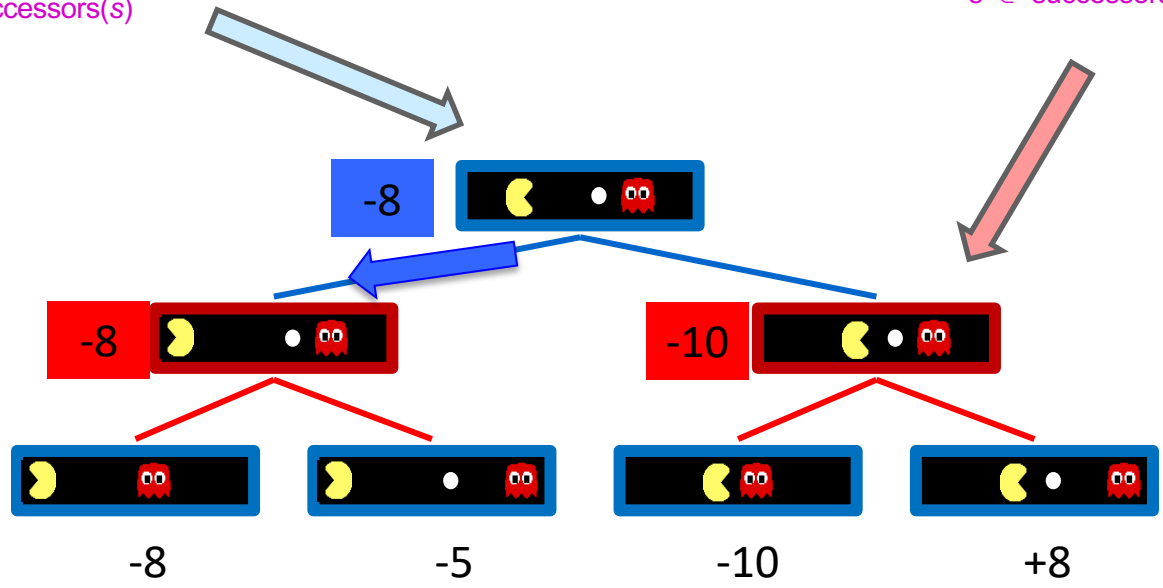
Small Pacman Example

MAX nodes: under Agent's control

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

MIN nodes: under Opponent's control

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

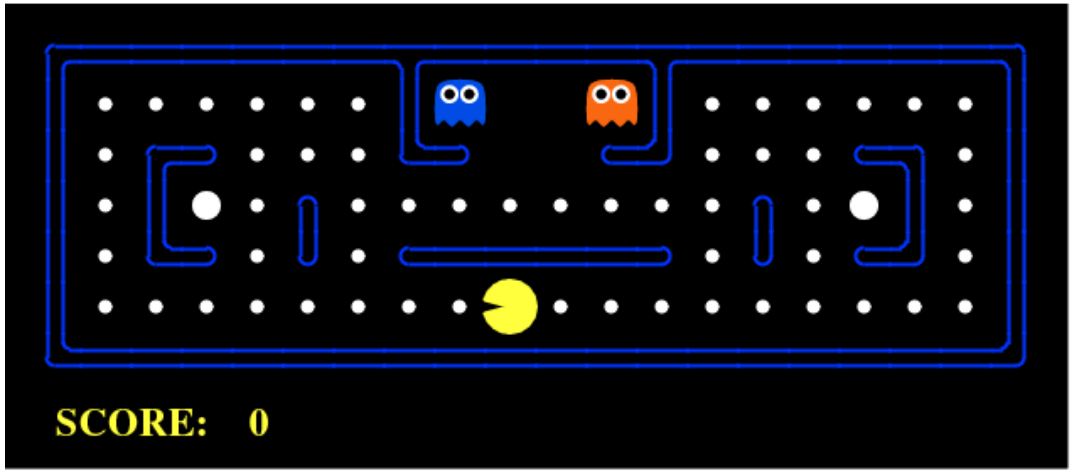


Terminal States:

$V(s)$ = known

HW3: Programming assignment

- PACMAN with ghosts



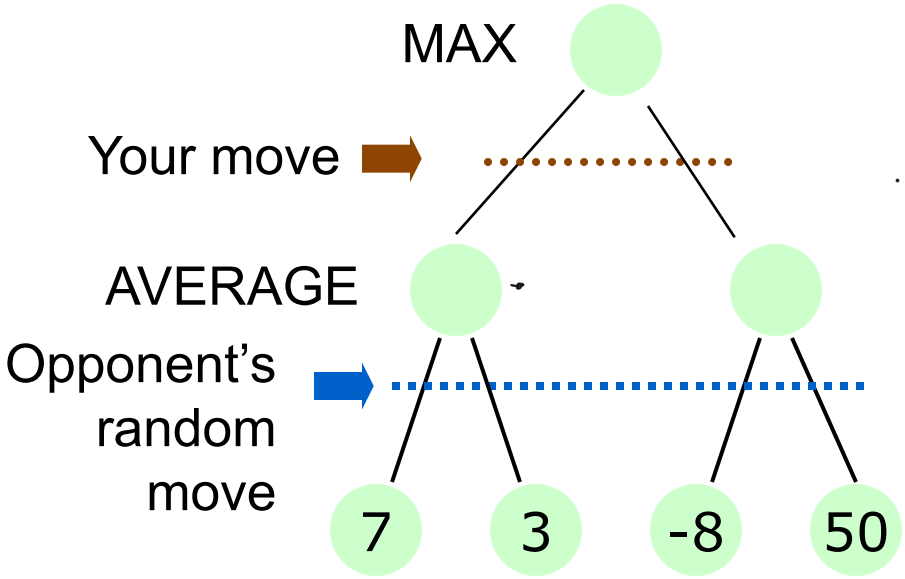
- You will generalize minimax search to multiple opponents
- You will exploit the benign opponents

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.

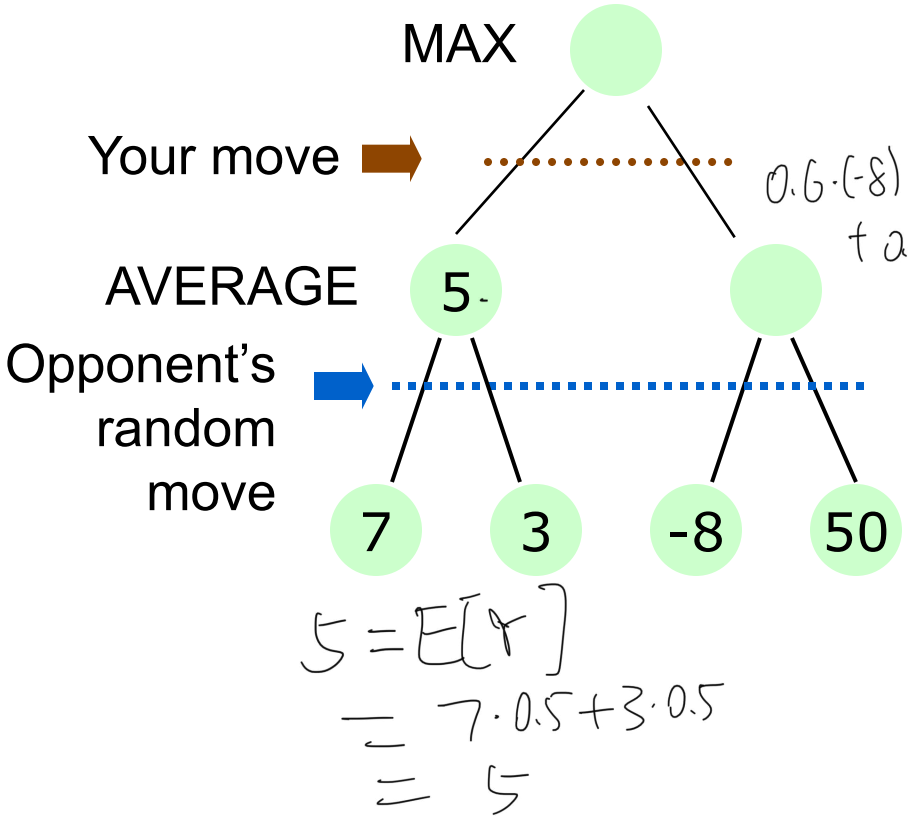
- Example of game of chance:
  - Slot machines
  - Tetris

Expectimax example



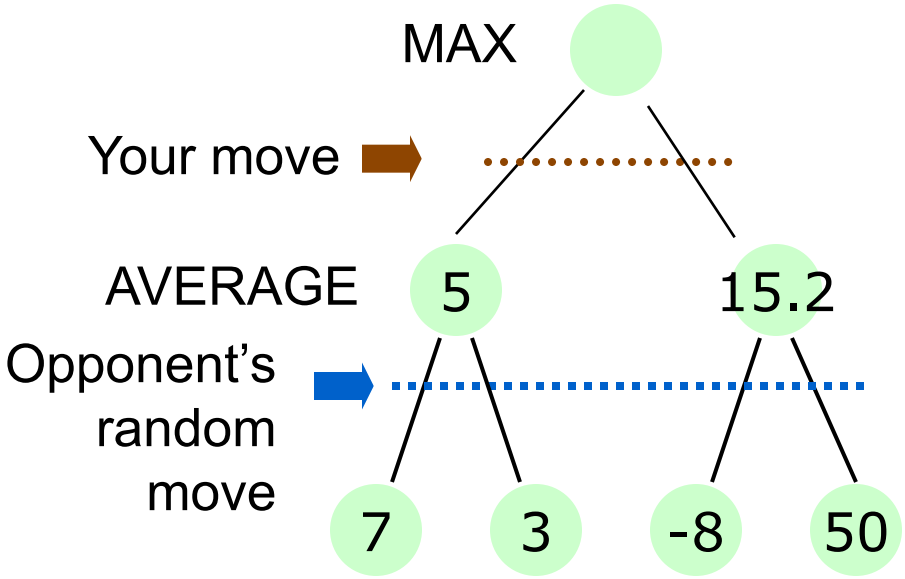
- Your opponent behave randomly with a given probability distribution,
- If you move left, your opponent will select actions with probability [0.5,0.5]
- If you move right, your opponent will select actions with [0.6,0.4]

Expectimax example



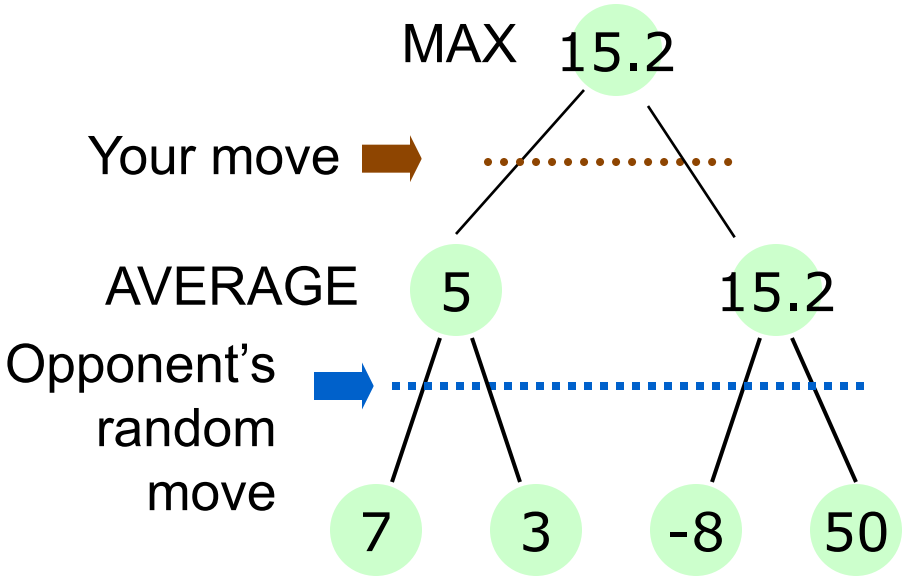
- Your opponent behave randomly with a given probability distribution,
- If you move left, your opponent will select actions with probability [0.5,0.5]
- If you move right, your opponent will select actions with [0.6,0.4]

Expectimax example



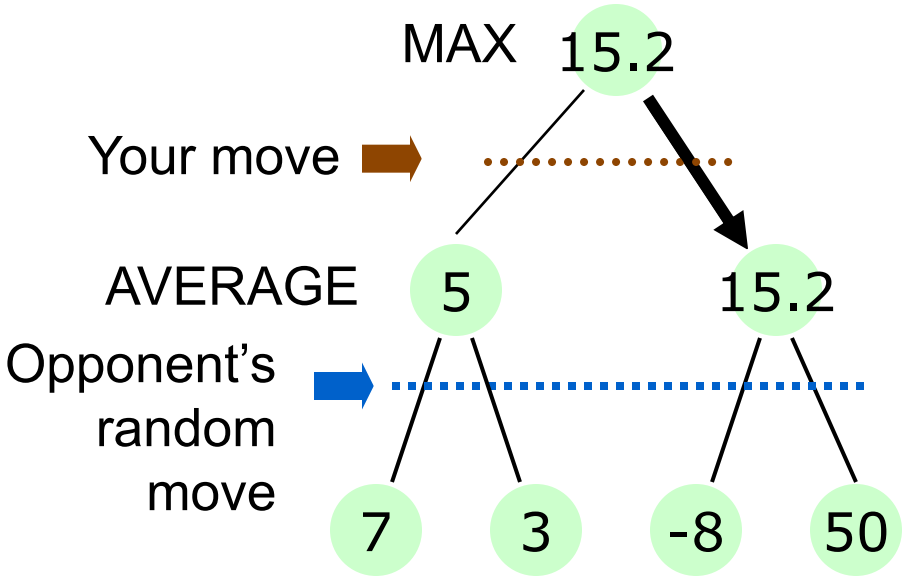
- Your opponent behave randomly with a given probability distribution,
- If you move left, your opponent will select actions with probability [0.5,0.5]
- If you move right, your opponent will select actions with [0.6,0.4]

Expectimax example



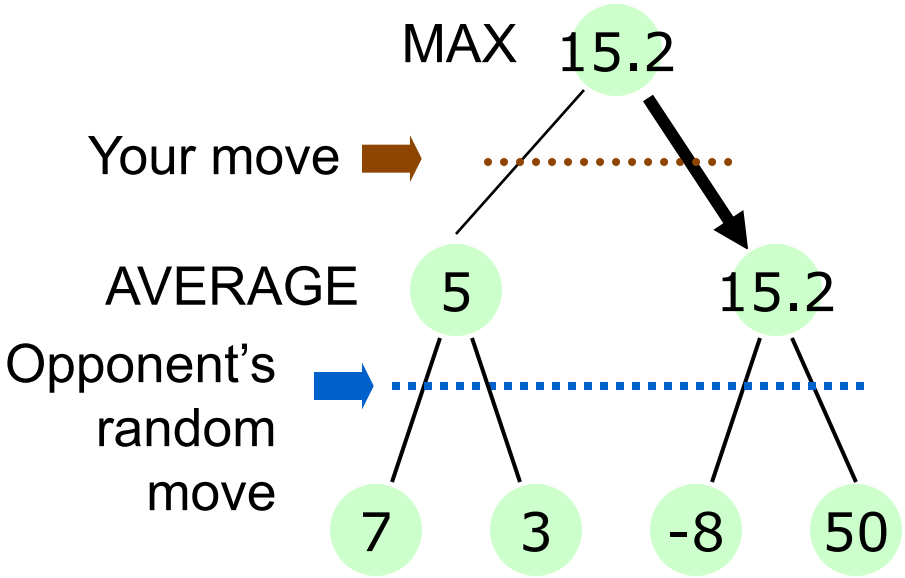
- Your opponent behave randomly with a given probability distribution,
- If you move left, your opponent will select actions with probability [0.5,0.5]
- If you move right, your opponent will select actions with [0.6,0.4]

Expectimax example



- Your opponent behave 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 behave 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. 5.7 (or in the "Historical notes" of the AIMA 4th Edition)
  - Deep blue
  - TD Gammon

- AlphaGo: https://www.nature.com/articles/nature16961

About the upcoming midterm

- **Topics:** Cover up to A*-Search and HW2. (not including Games / Adversarial search)
- **Format:** Take-home, open book, but **strictly** independent work.
- **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)
- **Format:** Take-home, open book, but **strictly** independent work.
- **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

- You have received an email with instructions.
  - Read carefully.

- You have 24 hours from 12:30 pm Thursday to 12:30 pm Friday, but most people should be able to finish all non-bonus questions within 1.5 hours.

- You will submit your scanned pages to Gradescope.
  - Make sure you write clearly and label the pages to questions.

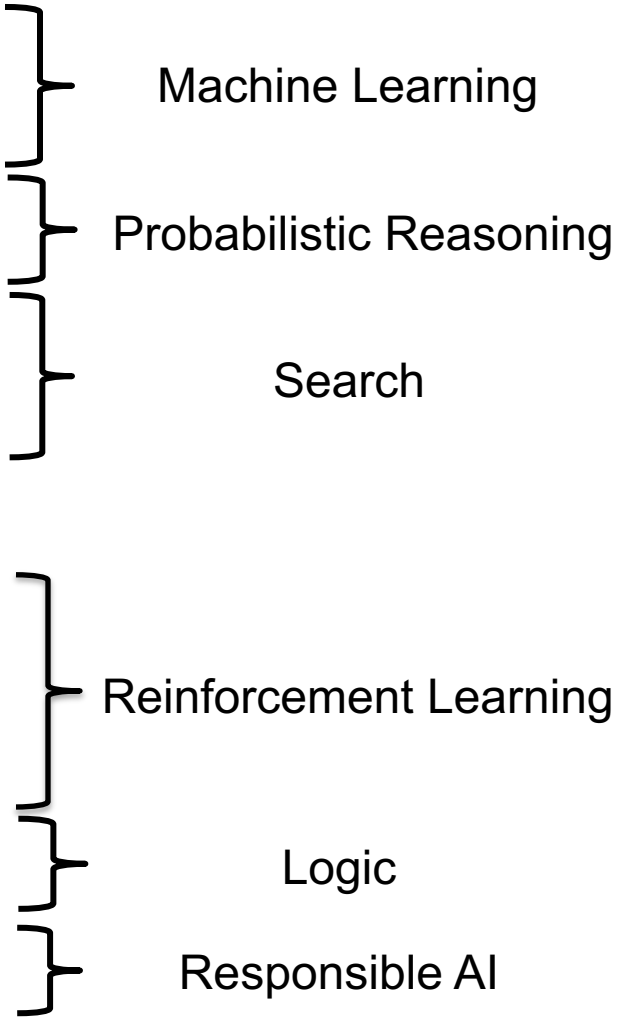
To help your preparation

- Discussion class for Homework 2 on Wednesday
  - The TA will play his recorded video in the first discussion section
  - Then Q&A.
  - You can watch the video any time you want.

- TAs moving office hours to before the midterm

Time to pause and take a look back!

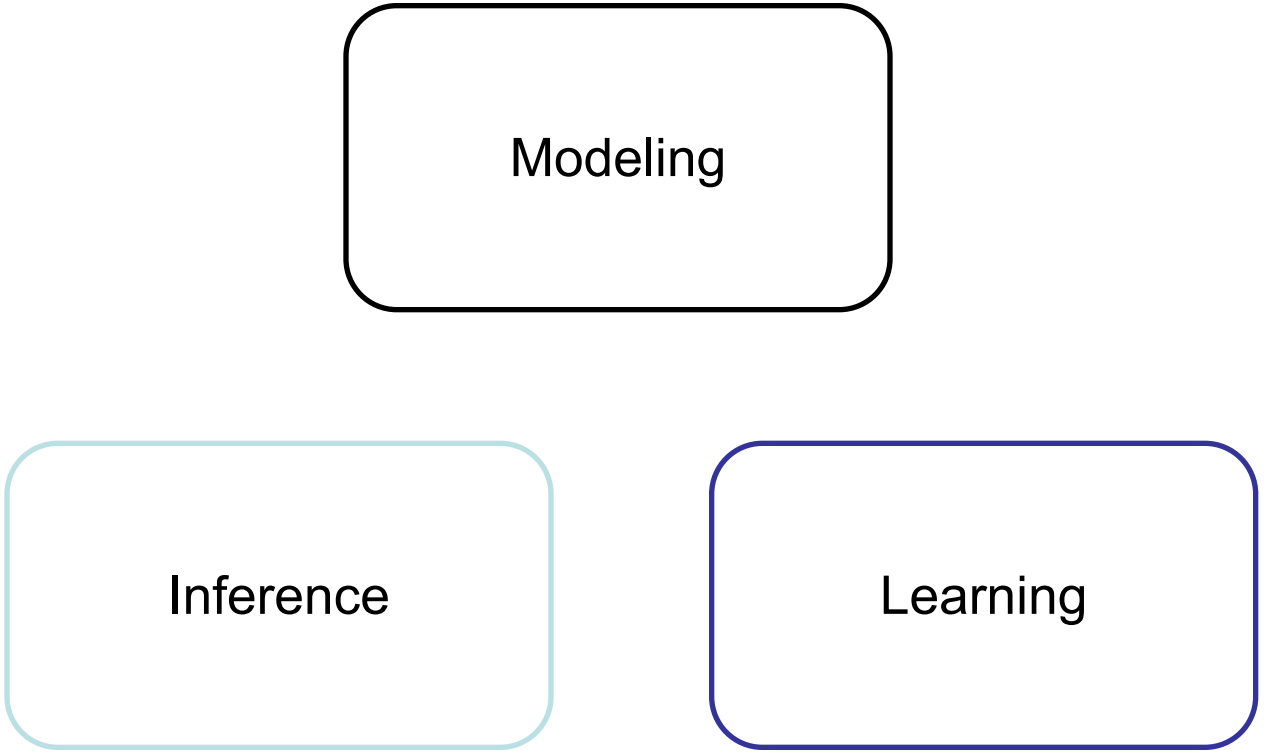
Week	Topic
1	Course Overview & Intelligent Agents
2	Machine Learning Machine Learning
2	Machine Learning Probabilistic Graphical Models
3	Probabilistic Graphical Models Search: Problem solving with search
4	Search: Search algorithms Search: Minimax search and game playing
5	Midterm Review **Midterm (take-home)**
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 (take-home)



Lecture 1: AI Overview & Agents

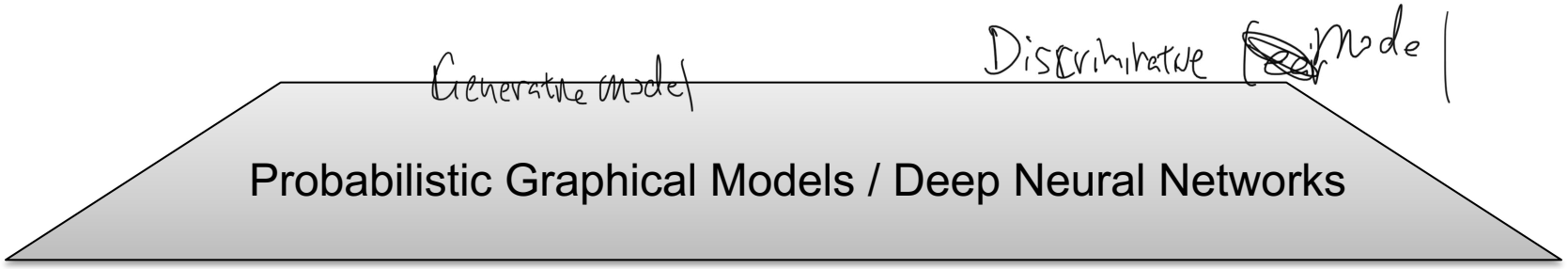
- AI for problem solving
- Rational agents
- Examples of AI in the real world
- Modelling-Inference-Learning Paradigm

New paradigm: Modeling-Inference-Learning



(Idea and example taken from Percy Liang's teachings)

Structure of the course



Classification / Regression
Bandits

Search
game playing

Markov Decision Processes
Reinforcement Learning

Logic, knowledge base
Probabilistic inference

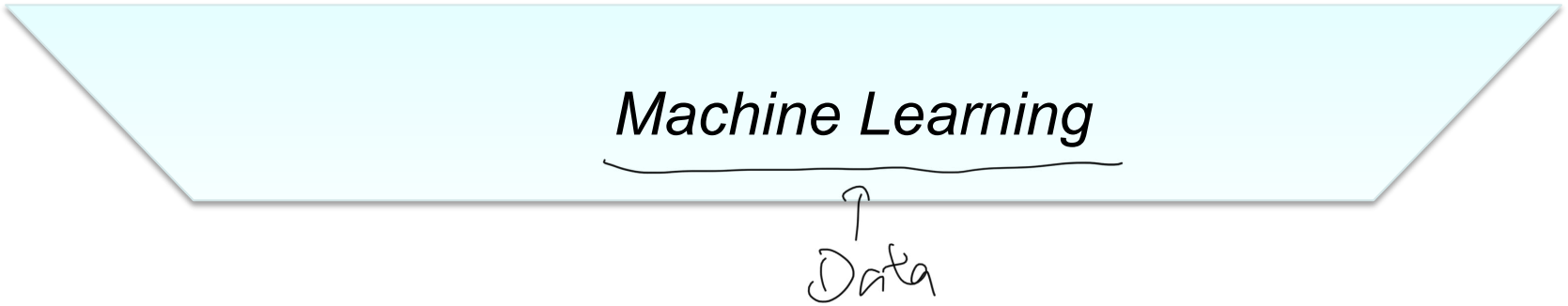
Reflex Agents

Planning Agents

Reasoning agents

Low-level intelligence

High-level intelligence



(Again this idea is adapted from Percy Liang's teachings)

Lecture 3-4: 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

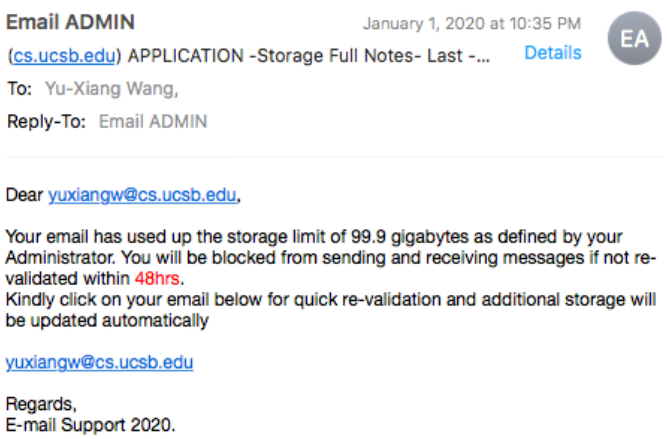
Contains hyperlinks

Proportion of misspelled words

1	0	0.0375	80

Whether the contact list

Length of the message



Email ADMIN January 1, 2020 at 10:35 PM

(cs.ucsb.edu) APPLICATION -Storage Full Notes- Last -... Details

To: Yu-Xiang Wang,

Reply-To: Email ADMIN

Dear 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

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

- Score(x) = $w_0 + w_1 * 1(\text{hyperlinks}) + w_2 * 1(\text{contact list}) + w_3 * \text{misspelling} + w_4 * \text{length}$

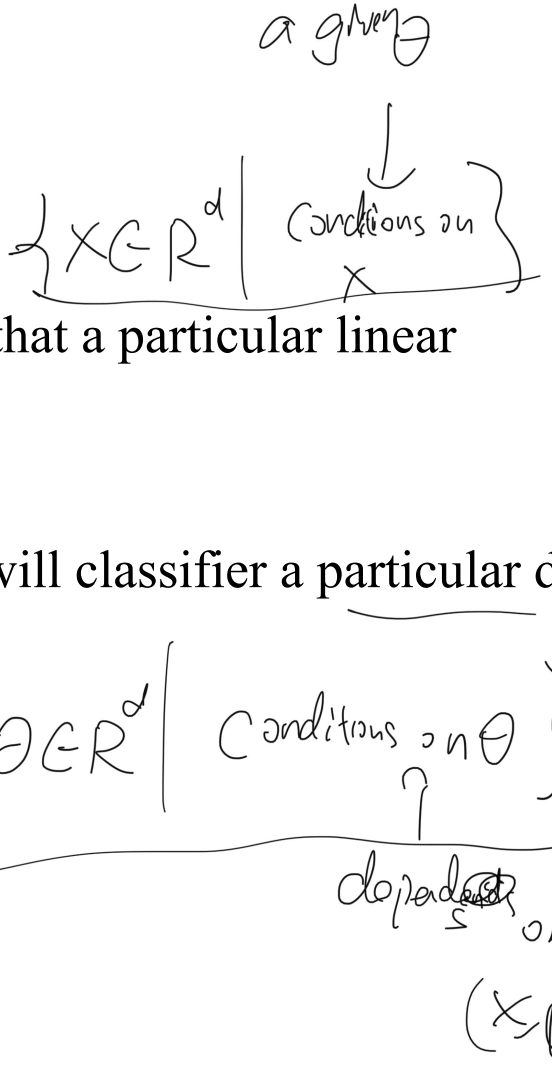
- A linear classifier: h(x) = 1 if Score(x) > 0 and 0 otherwise.

- Question: What are the "free-parameters" in a linear classifier?
  - If we redefine $\mathcal{Y} = \{-1, 1\}$
  - A compact representation:

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

Linear classifier

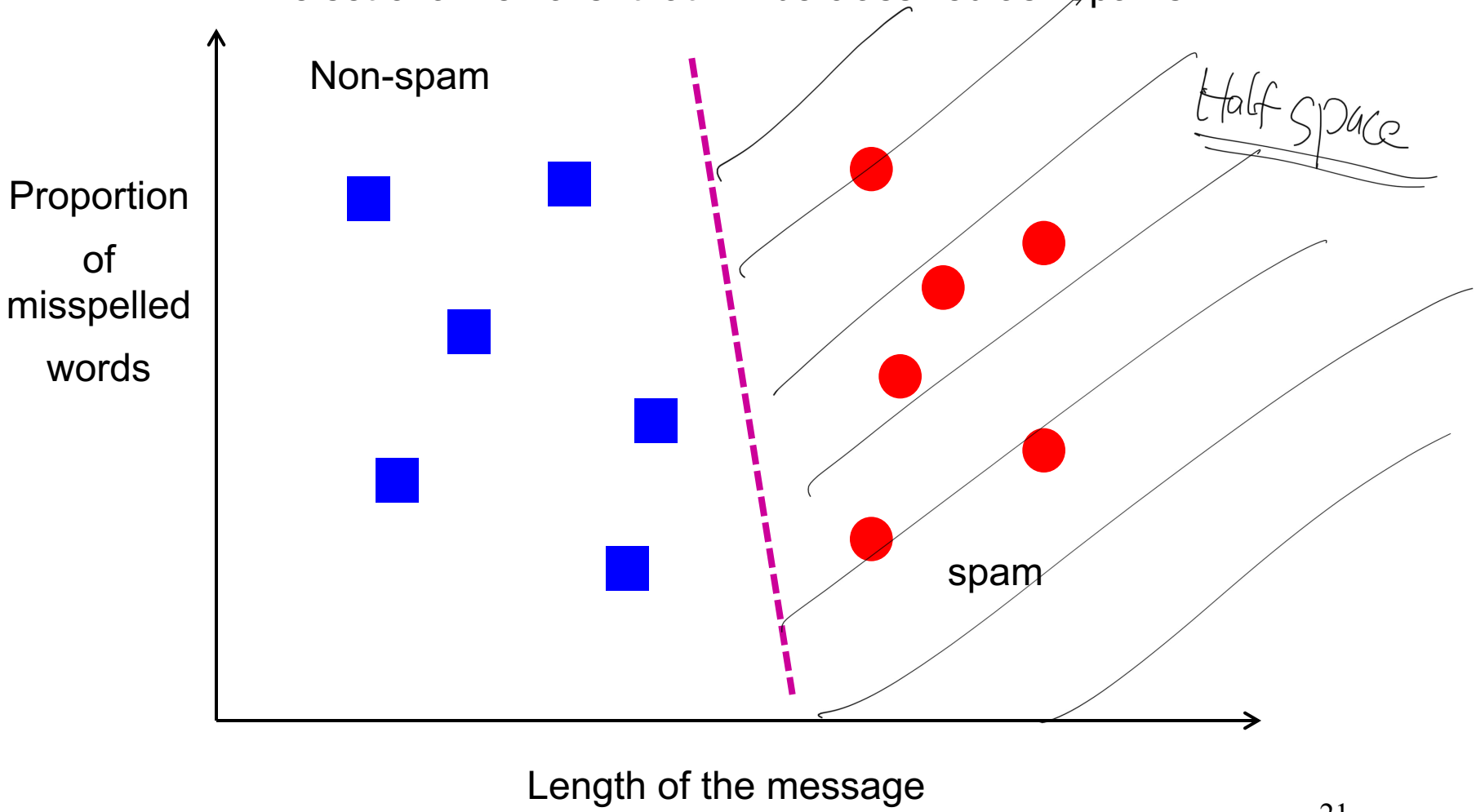
- Think geometrically
  - What is the set of all feature vectors that a particular linear classifier will classify as "Spam"?
  - What is the set of all classifiers that will classifier a particular data point to be "Spam"?



Geometric view: Linear classifier are "half-spaces"!

$\{ x \mid w_0 + w_1 * x1 + w_2 * x2 + w_3 * x3 + w_4 * x4 > 0 \}$

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



Just "relax": relaxing a hard problem into an easier one

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

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

$w_{t+1} = w_t - \eta_t g_t$

GD $g_t = \nabla \left(\frac{1}{n} \sum_{i=1}^{n} \ell(w_t^T x_i, y_i) \right)$

SGD { ① randomly sample i in Unif$(1, 2, \ldots, n)$
$g_t = \nabla \ell(w_t^T x_i, y_i)$

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: $(y - S_w(x))^2$

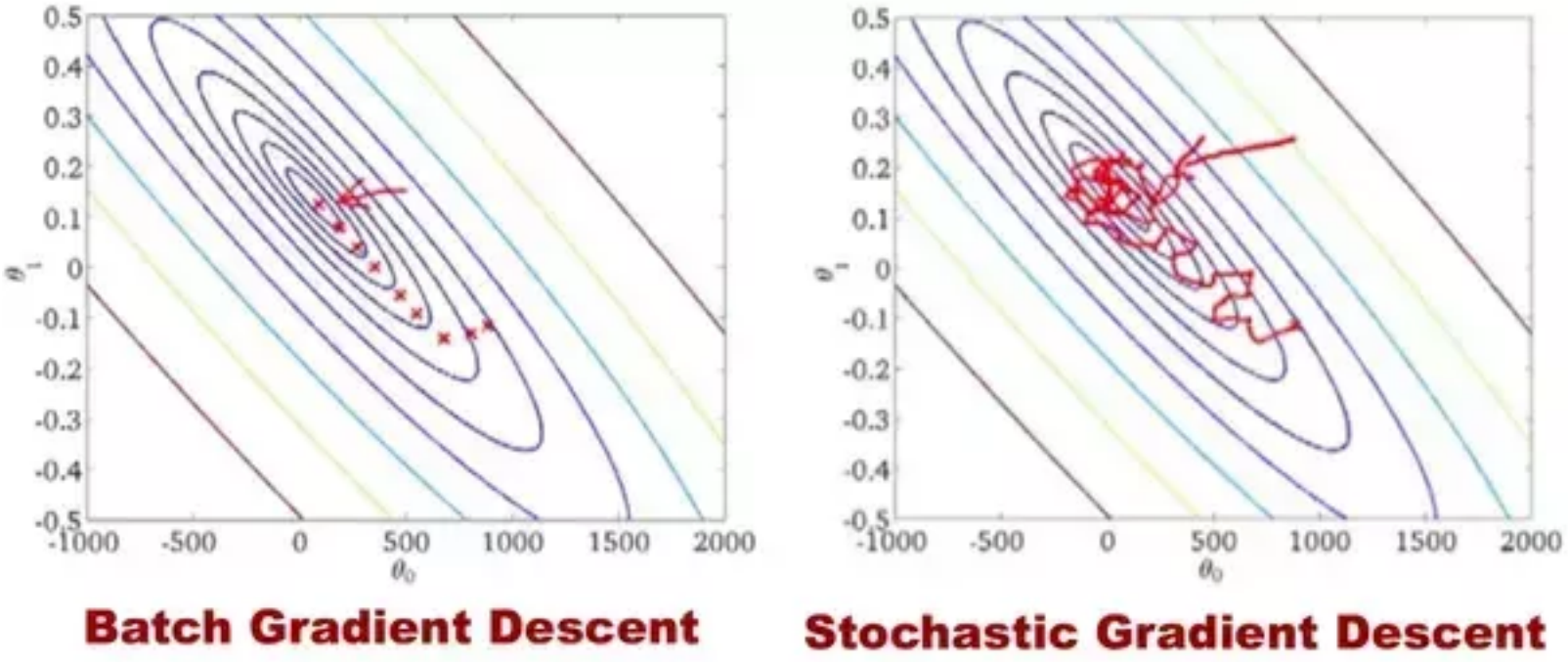
Loss functions and surrogate losses

- 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



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

Soft. argmax

$$\nabla \ell(w, (x_i, y_i)) = \frac{\exp(-y_i \cdot x_i^T w)}{1 + \exp(-y_i \cdot x_i^T w)} (-y_i \vec{x_i})$$

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

- Example: Binary logistic loss

$$\ell_{(x,y)}(\theta) = \log\left(1 + e^{-yx^T\theta}\right), \quad y \in \{-1, 1\}, \; x \in \mathbb{R}^d$$

$$\nabla \ell(\theta) = \begin{bmatrix} \frac{\partial \ell(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial \ell(\theta)}{\partial \theta_d} \end{bmatrix}$$

$$\frac{\partial \ell(\theta)}{\partial \theta_j} = \frac{1}{1 + e^{-yx^T\theta}} \cdot e^{-yx^T\theta} \cdot (-yx_j), \quad x^T\theta = \sum_{j=1}^{d} x_j \cdot \theta_j$$

$$\nabla \ell(\theta) = \frac{e^{-yx^T\theta}}{1 + e^{-yx^T\theta}} \cdot \vec{x}$$

- Example: Cross-entropy loss

$$\Theta = [\theta_1, \cdots \theta_k] \in \mathbb{R}^{d \times k}, \; \theta_j \in \mathbb{R}^d$$

$$\ell(\Theta) = -\sum_{i=1}^{k} y[i] \cdot \log(\hat{y}(i)) = -\sum_{i=1}^{k} y(i) \cdot \log\left(\frac{e^{\theta_i^T x}}{\sum_{\ell=1}^{k} e^{\theta_\ell^T x}}\right) = -\sum_{i=1}^{k} y(i) \cdot \theta_i^T x + \underbrace{\sum_{i=1}^{k} y(i)}_{=1 \text{ always}} \log \sum_{\ell=1}^{k} e^{\theta_\ell^T x}$$

$$\nabla_{\theta_j} \ell(\Theta) = -y(j) \cdot \vec{x} + \left(\sum_{i=1}^{k} y(i)\right) \frac{1}{\sum_{\ell=1}^{k} e^{\theta_\ell^T x}} \cdot e^{\theta_j^T x} \cdot \vec{x}$$

all other summands are constant w.r.t. θ_j; $\sum_{i=1}^{k} y(i) = 1$; $\frac{1}{\sum_{\ell=1}^{k} e^{\theta_\ell^T x}} \cdot e^{\theta_j^T x} = \hat{y}(j)$ (soft-argmax, i.e. $\hat{y}(j)$)

$$= \vec{x} \cdot (\hat{y}(j) - y(j))$$

Putting things together:

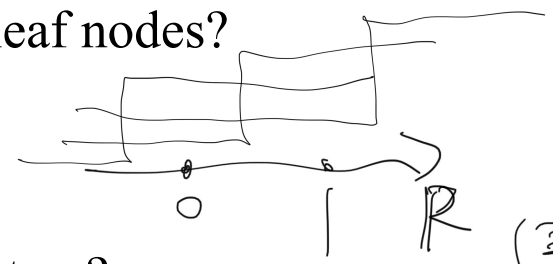
$$\nabla \ell(\Theta) = \vec{x} \cdot [\hat{y}(1) - y(1), \cdots, \hat{y}(k) - y(k)]$$

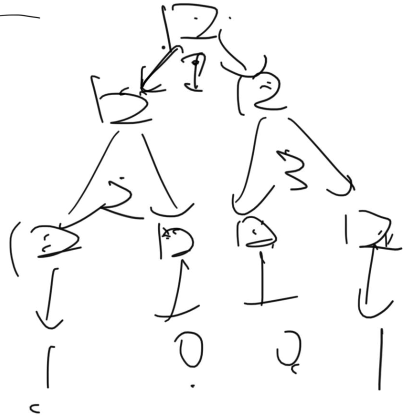
$$= \vec{x} \cdot (\hat{y} - y)^T$$

dimension check: $d \times 1 \;\; 1 \times k \Rightarrow d \times k$

Quiz on Piazza: Learning a decision tree

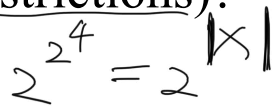
- Free parameters:
  - Which feature(s) to use when branching branch?
  - How to branch? Thresholding? Free threshold?
  - Which label to assign at leaf nodes?





- Hyperparameters:
  - Max height of a decision tree?
  - Number of parameters the tree can use in each

- **Question**: Consider a problem with **4 binary features**.
  - How many decision trees of **3 layers** are there? If each decision uses only one feature? (you may repeat features)
  - How many possible feature vectors are there?
  - How many classifiers are there (without restrictions)?



Lecture 5-7: Probabilities and BayesNet

- Modelling the world with a joint probability distribution
  - Number of 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

Fully Independent

$P(X_1, X_2, \ldots, X_N)$

$= P(X_1)\, P(X_2) \ldots P(X_N)$

$O(N)$

Fully general

$P(X_1, X_2, \ldots, X_N)$

$O(e^N)$

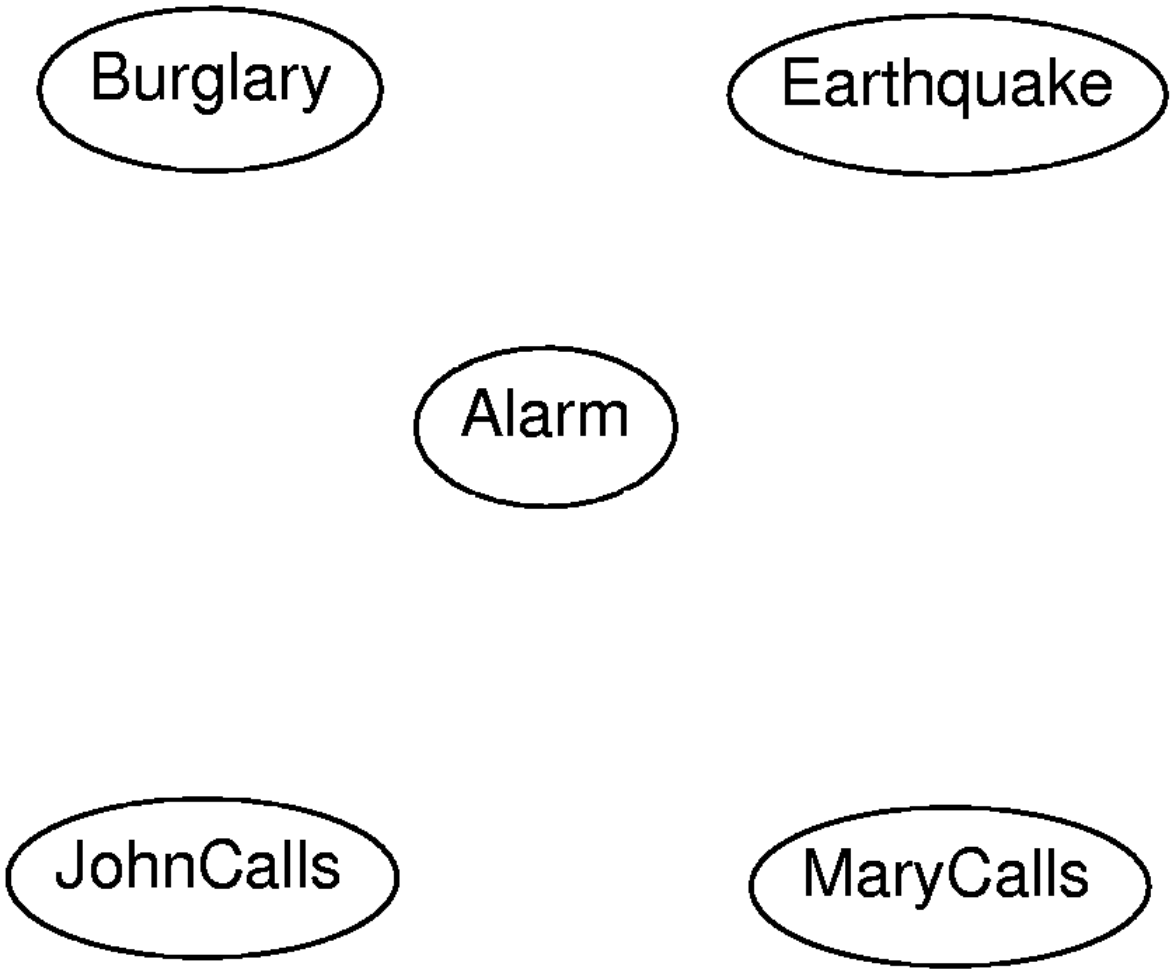
Space / computation efficiency

Expressiveness

Idea:

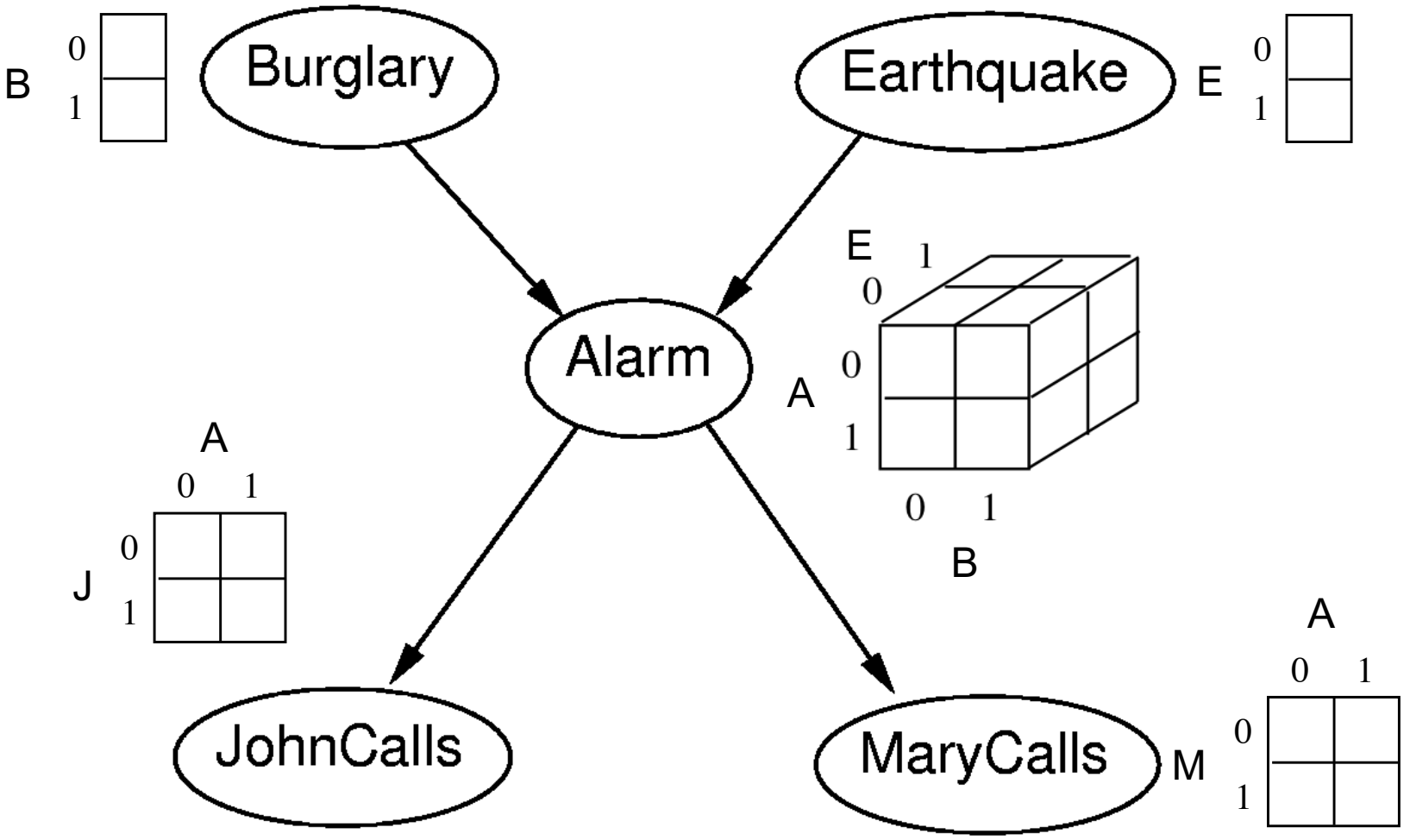
1. Independent groups of variables?
2. **Conditional independences?**

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



Links and CPTs?

What are the CPTs? What are their dimensions?

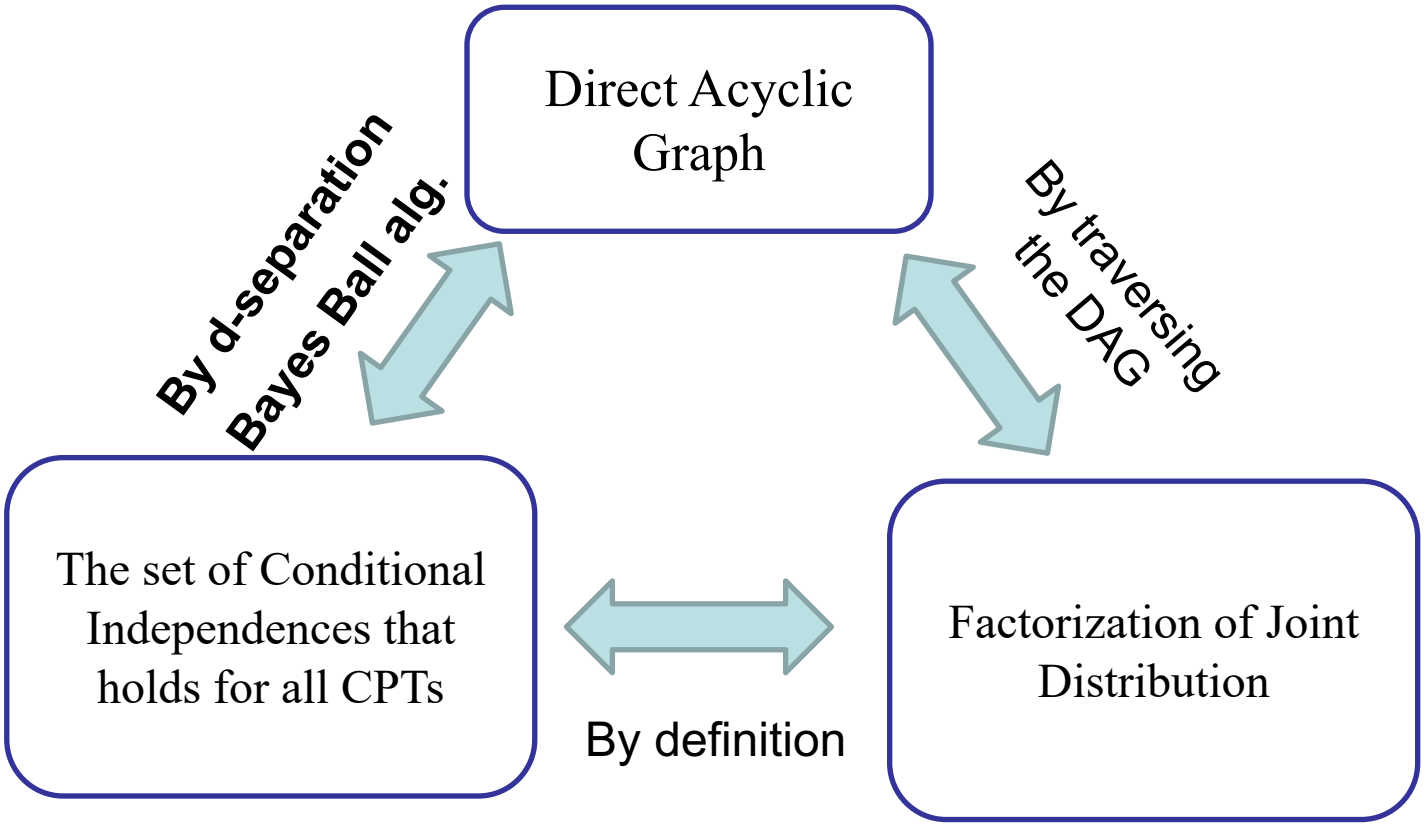


Question: How to fill values into these CPTs?

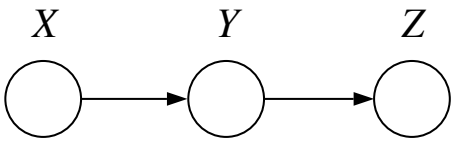
Ans: Specify by hands. Learn from data (e.g., MLE).

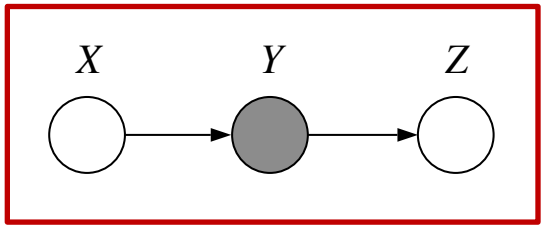
Big question: Is there a general way that we can answer questions about conditional independences by just inspecting the graphs?

- Turns out the answer is "Yes!"



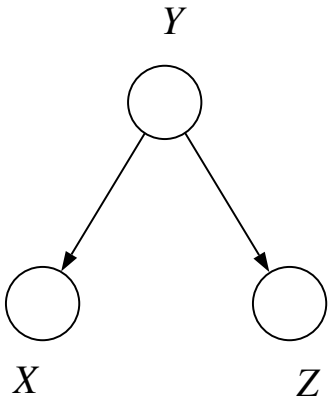
d-separation in three canonical graphs

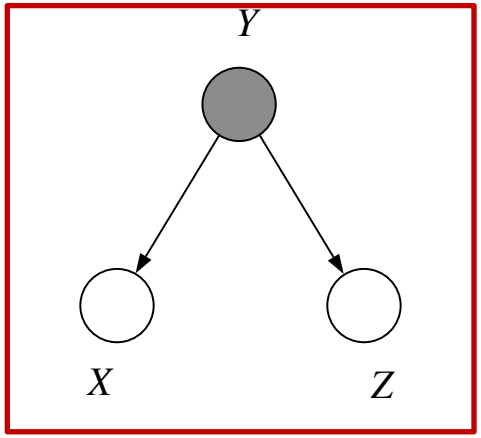




X ⊥ Z | Y

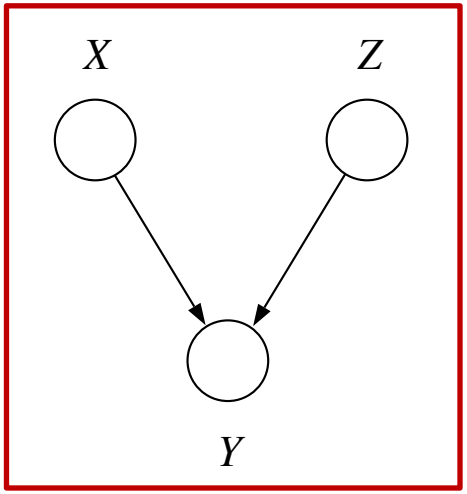
"X and Z are d-separated by the observation of Y."

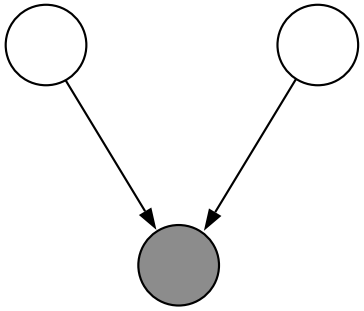




X ⊥ Z | Y

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

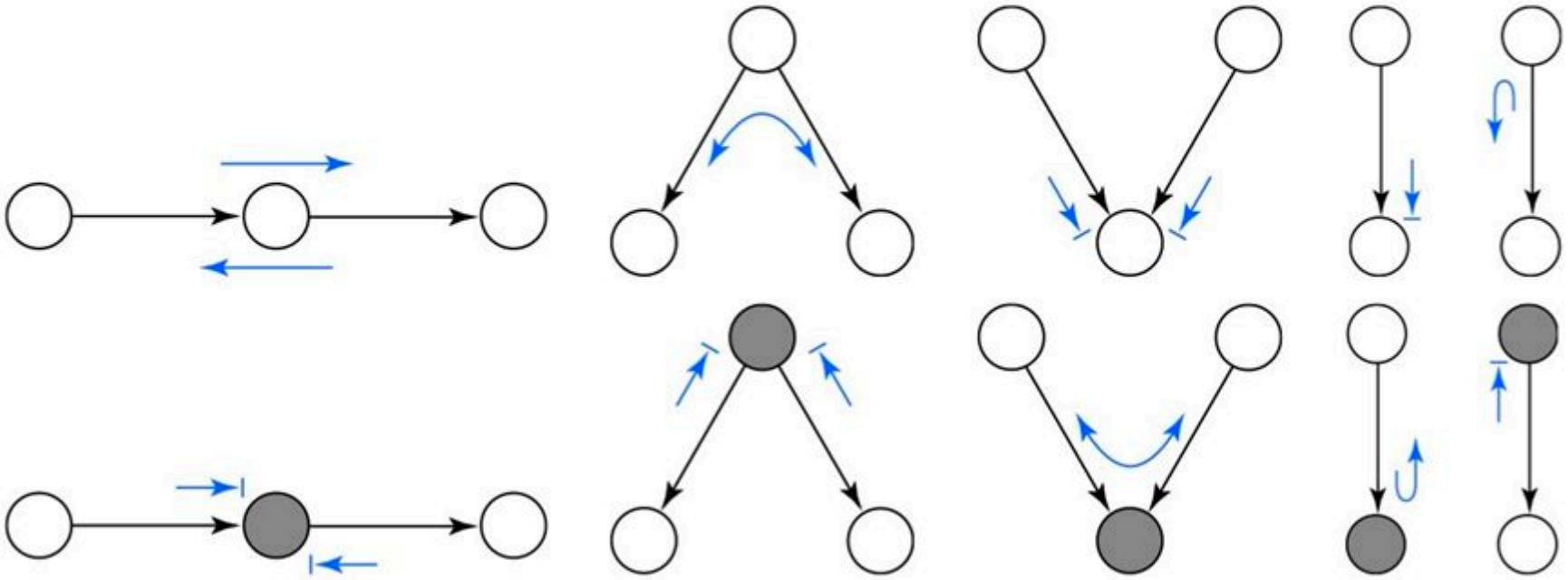




X ⊥ Z | Y

"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

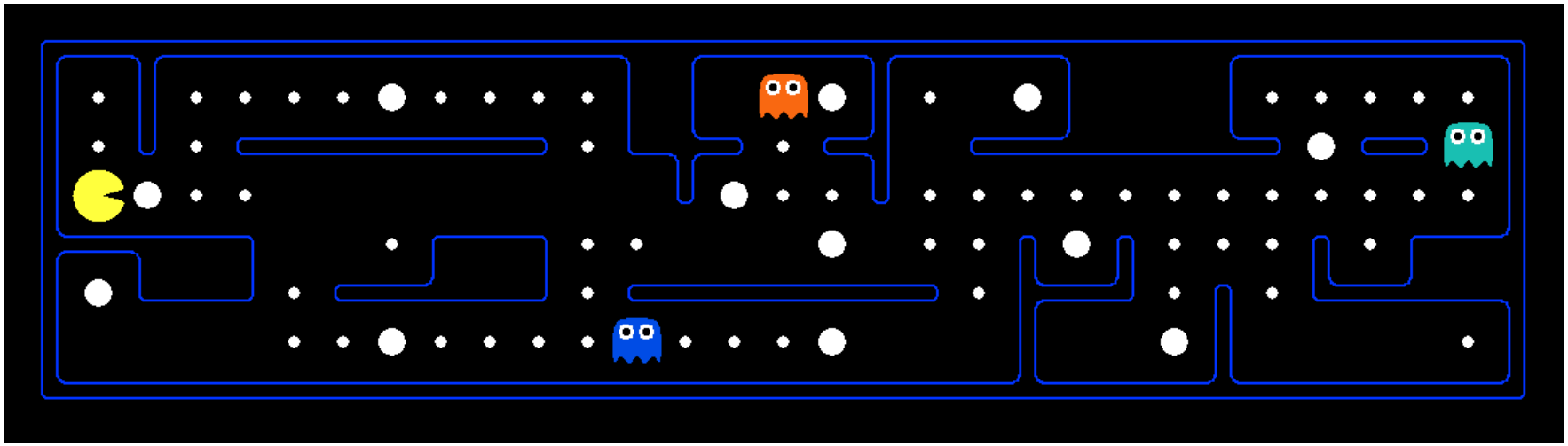
- Heuristic search
  - When does it work?

Problem Formulation and Search

- Problem formulation
  - State-space description **< {S}, S_0, {S_G}, {O}, {g} >**
    - **S**: Possible states
    - **S_0**: Initial state of the agent
    - **S_G**: Goal state(s)
      - Or equivalently, a goal test **G(S)**
    - **O**: Operators O: {S} => {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_i 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

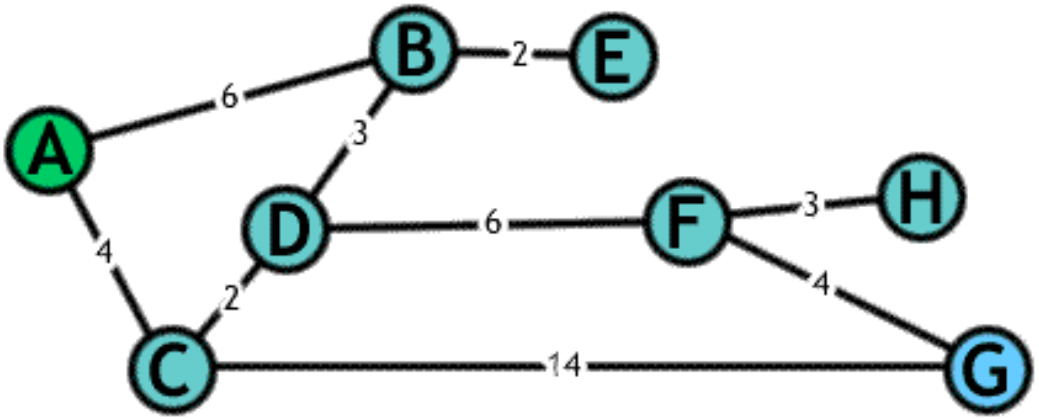
- The goal is to eat all pellets as quickly as possible while staying alive. Eating the "Power pellet" will allow the pacman to eat the ghost.



- Think about how to formulate this problem.

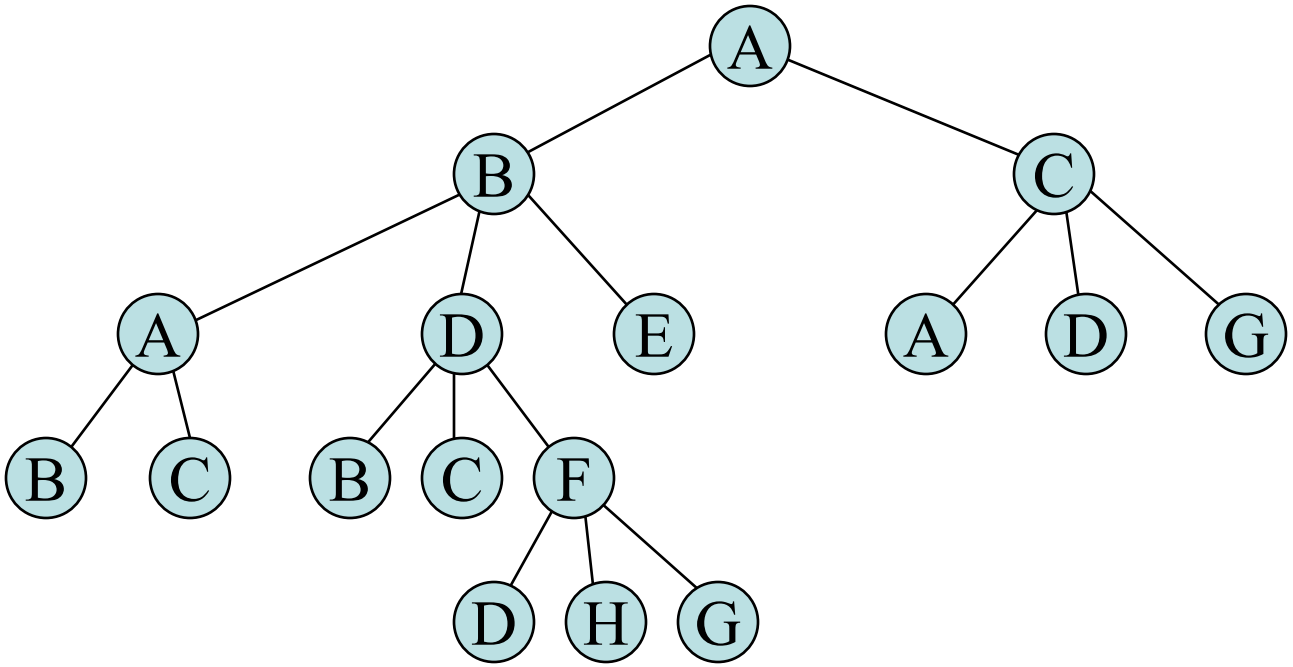
State-Space Diagrams

- 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[#]	Yes[#&]	No	No	Yes[#]	Yes[#+]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/e \rfloor})$	$O(b^m)$	$O(b^l)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/e \rfloor})$	$O(bm)$	$O(bl)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[$]	Yes	No	No	Yes[$]	Yes[$+]

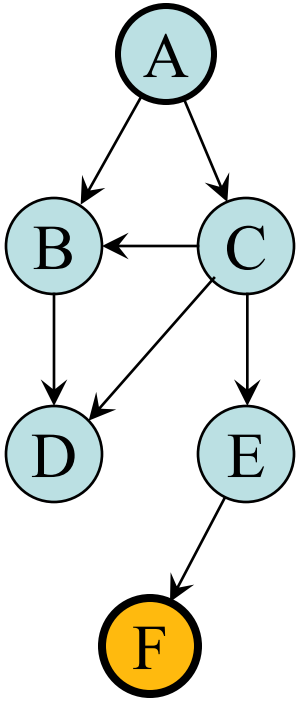
b: Branching factor
d: Depth of the shallowest goal
l: Depth limit
m: Maximum depth of search tree
e: The lower bound of the step cost

[#]: Complete if *b* is finite
[&]: Complete if step cost >= *e*
[$]: Optimal if all step costs are identical
[+]: If both direction use BFS

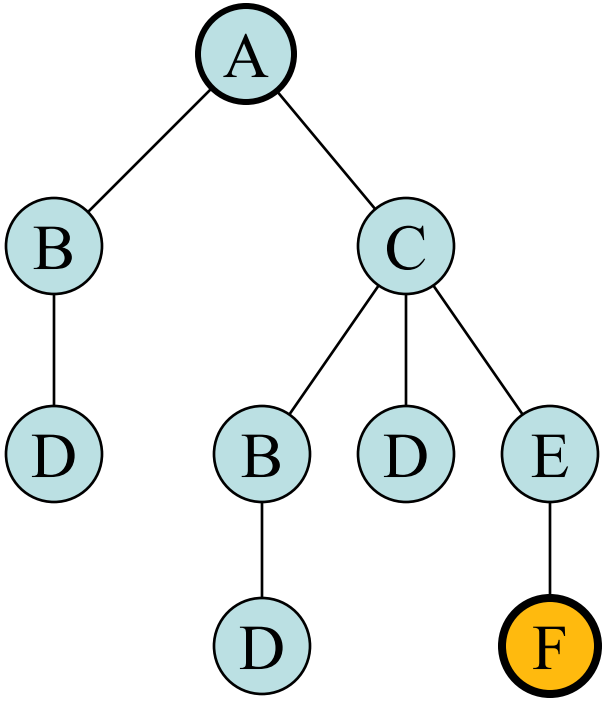
(Section 3.4.7 in the AIMA book.)

Example

State space graph



Search tree



Queue

(A)

(B C)

(C D)

(D B D E)

(B D E)

(D E D)

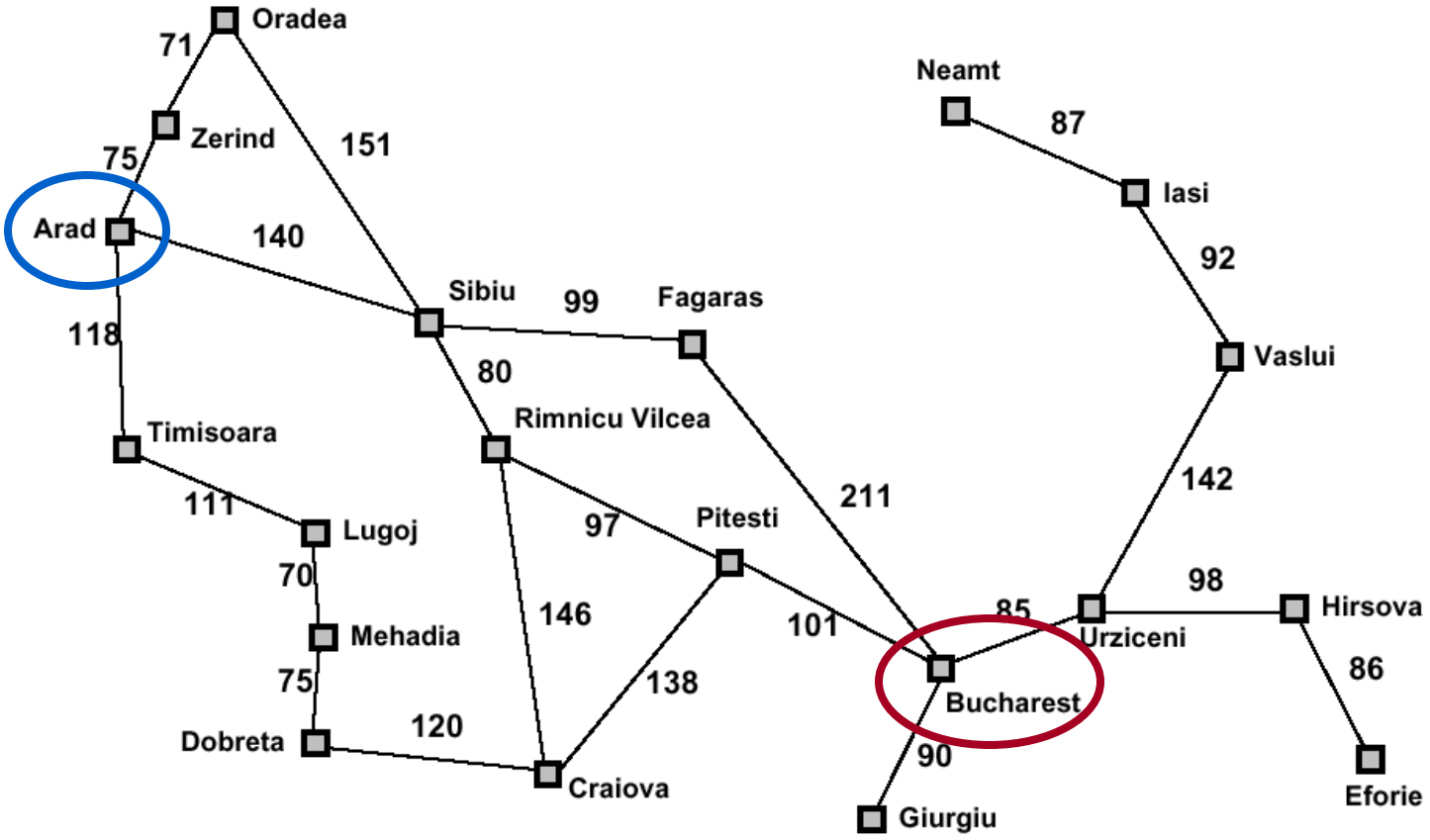
(E D)

(D F)

(F)

( )

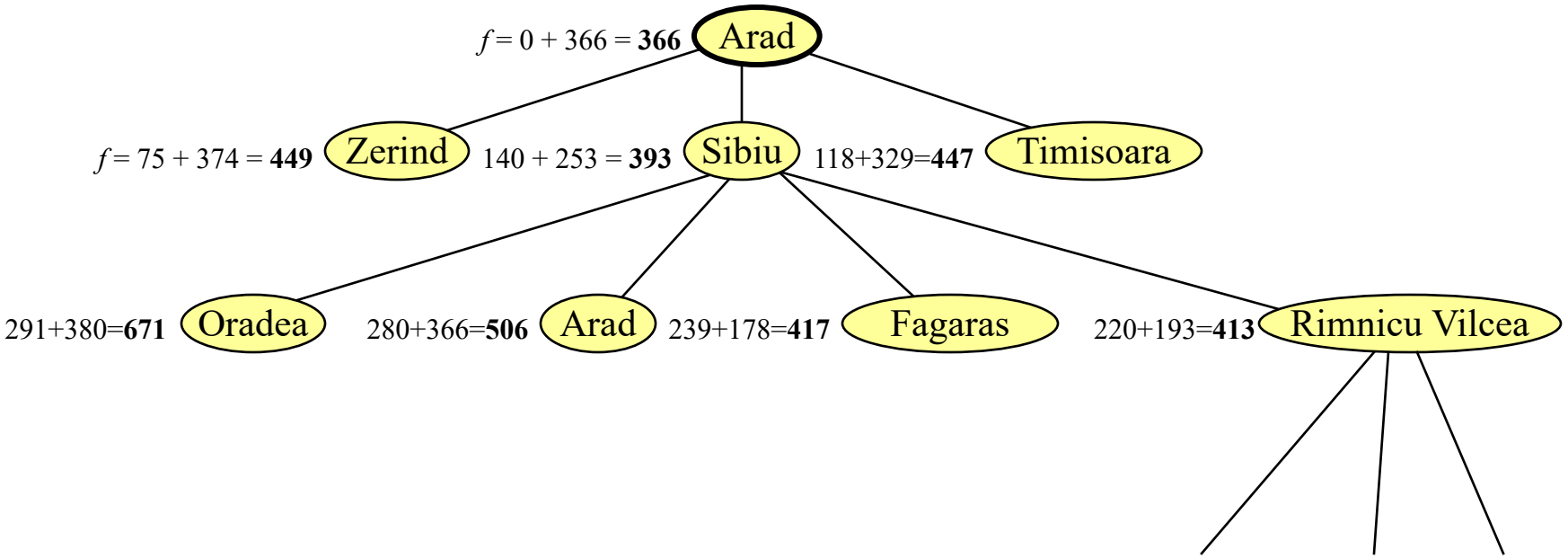
A* Example



Straight–line distance to Bucharest	
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

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

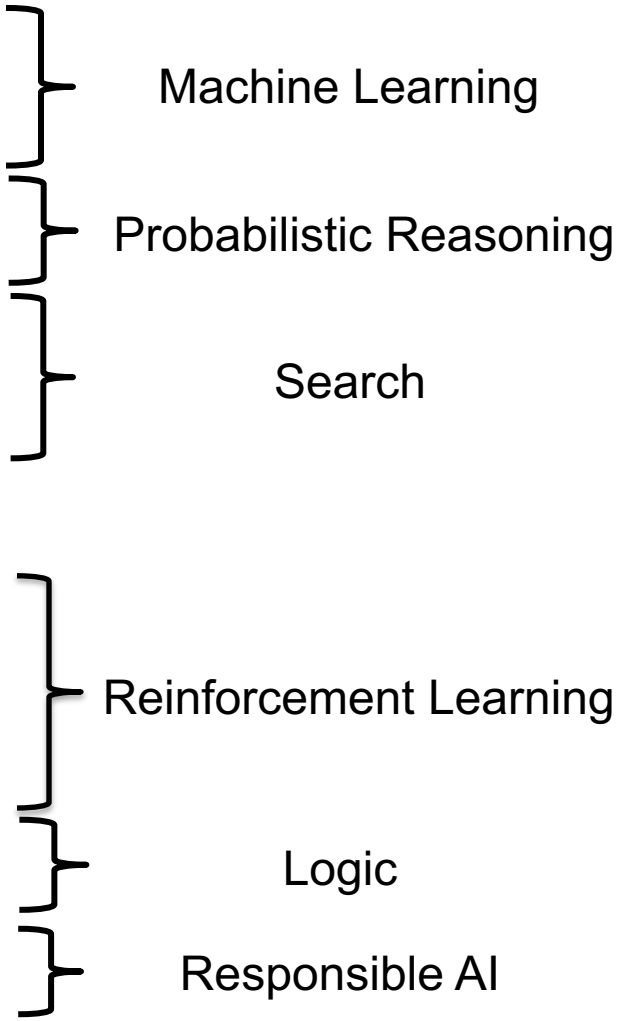
A* Example



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 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 **Midterm (take-home)**
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 (take-home)



Optimization perspective of AI

Optimization perspective of AI

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

Optimization perspective of AI

- A rational agent $\max_{a_1,\ldots,a_T} \text{Utility}(a_1, \ldots, a_T)$
  - **Modelling tools:** Features / Hypothesis, 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 $\max_{a_1,\ldots,a_T} \text{Utility}(a_1, \ldots, a_T)$
  - **Modelling tools:** Features / Hypothesis, 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

- 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!