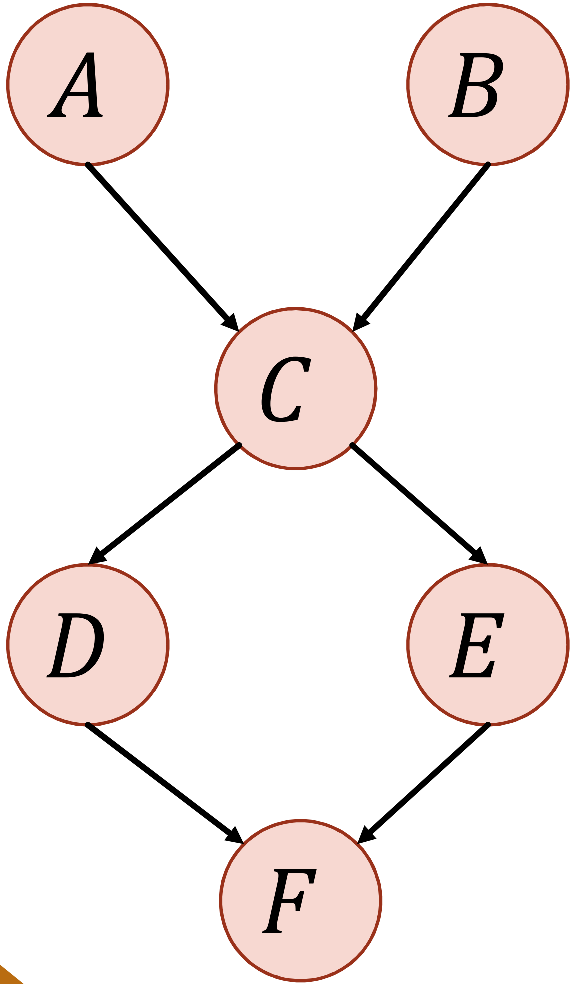
Bayesian Networks

... continued

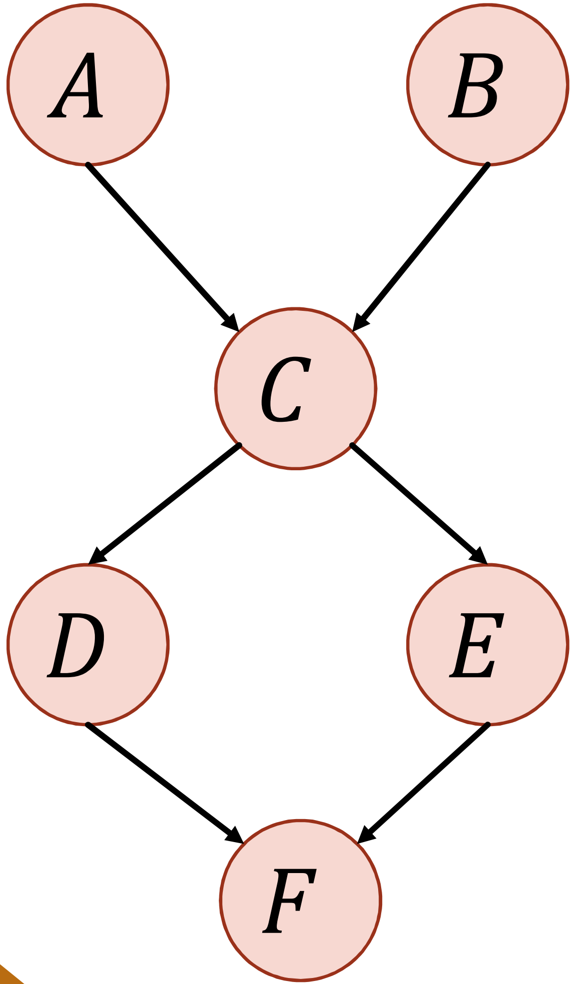
Bayesian Networks



A good way to factor joint distributions of random variables:

A variable is only conditionally dependent on its parents.

Conditional Independence in BN



Given variables X, Y and **known variables**

$$\mathcal{E} = \{E_1, \ldots, E_k\}$$

are X and Y independent given knowledge of $\mathcal{E}$?

Conditional Independence in BN

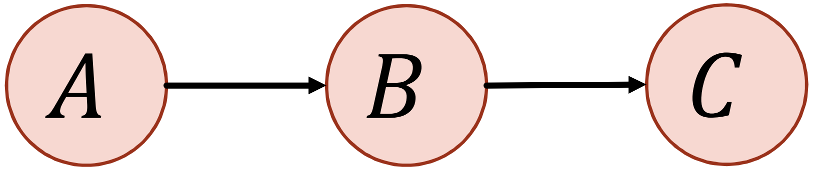
Given variables X, Y and **known variables**
$$\mathcal{E} = \{E_1, \dots, E_k\}$$
are X and Y independent given knowledge of $\mathcal{E}$?

Can be shown

- using algebra (annoying and tedious):
$$\Pr[\,X \mid \mathcal{E}\,] = \cdots = \Pr[\,X \mid \mathcal{E}, Y\,]$$
- via counterexample (computing via the CPTs)

Can we show that two nodes are **necessarily independent?**

Causal Chains

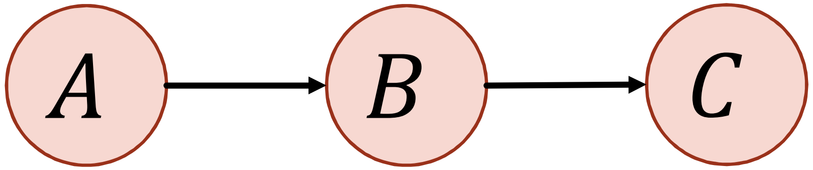


"Rain (A) causes traffic (B) which causes me to be late (C)"

Question: are A and C **necessarily independent**?

Question: are A and C **conditionally independent**, given B?

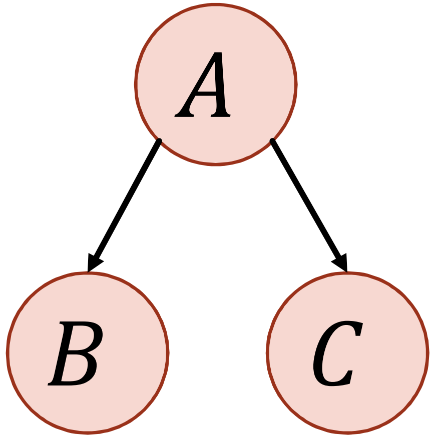
Causal Chains



$$\Pr[\,C \mid A, B\,] = \frac{\Pr[A \wedge B \wedge C]}{\Pr[A \wedge B]} = \frac{\Pr[A]\Pr[\,B \mid A\,]\Pr[\,C \mid B\,]}{\Pr[A]\Pr[\,B \mid A\,]} = \Pr[\,C \mid B\,]$$

$\Pr[\,C \mid A, B\,] = \Pr[\,C \mid B\,]$: given B, knowing A does not update my beliefs on C!

Common Cause

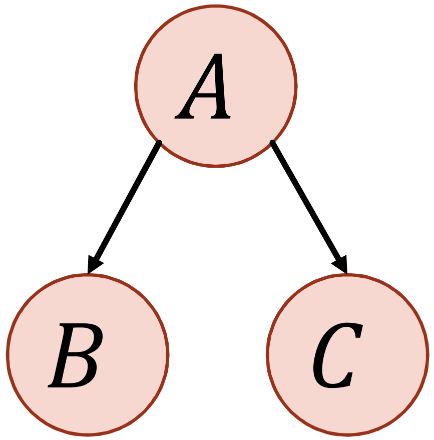


"Batman (A) catches the Joker (B) and Bane (C)"

Question: are B and C **necessarily independent**?

Question: are B and C **conditionally independent**, given A?

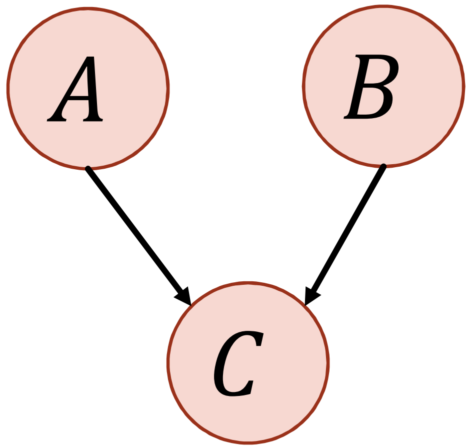
Common Cause



$$\Pr[\,B \mid A, C\,] = \frac{\Pr[A \wedge B \wedge C]}{\Pr[A \wedge C]} = \frac{\Pr[A]\,\Pr[\,B \mid A\,]\,\Pr[\,C \mid A\,]}{\Pr[A]\,\Pr[\,C \mid A\,]} = \Pr[\,B \mid A\,]$$

$\Pr[\,B \mid A, C\,] = \Pr[\,B \mid A\,]$: given A, knowing C does not update my beliefs on B!

Common Effect

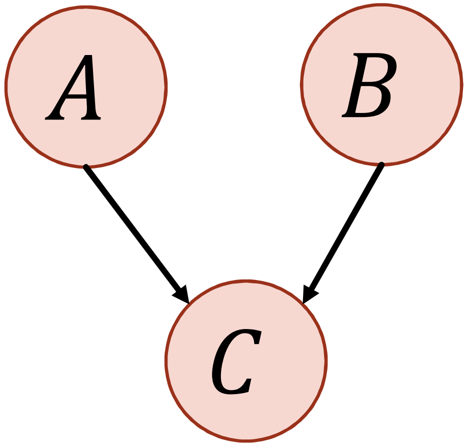


"The Joker (A) and Bane (B) could both rob the bank (C)"

Question: are A and B **necessarily independent**?

Question: are A and B **conditionally independent**, given C?

Common Effect



Observing an effect makes two causes dependent

- I know that the bank was robbed ($C = 1$)
- It could be either the Joker or Bane.
- If I know the Joker didn't do it –my belief about Bane doing it is higher!

$$\Pr[\, A \mid C, B \,] \neq \Pr[\, A \mid C \,]$$

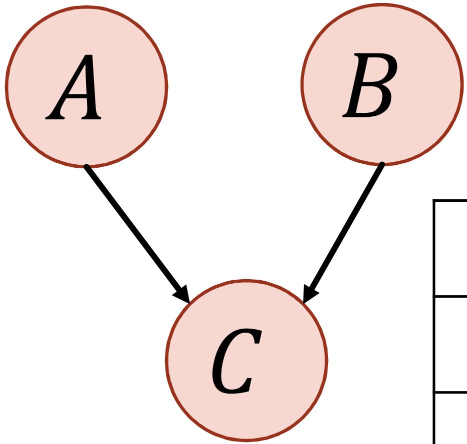
but

$$\Pr[\, A \mid B \,] = \Pr[A]$$

It’s All About the CPTs

$\Pr[A] = 0.5$

$\Pr[B] = 0.5$



$A =$	$B =$	$\Pr[\, C \mid A, B \,] =$
1	1	1
1	0	1
0	1	1
0	0	0

$$\Pr[A = 1] = \Pr[\, A = 1 \mid B = 0 \,] = 0.5$$

but

$$\Pr[\, A = 1 \mid B = 0, C = 1 \,] = 1$$

General Case – d Separation

Given variables X, Y and **known variables**

$$\mathcal{E} = \{E_1, \dots, E_k\}$$

are X and Y **surely** independent given $\mathcal{E}$?

Idea: any general graph can be broken down into the three cases described above, to determine conditional independence of X, Y given knowledge of $\mathcal{E}$.

General Case – d Separation

Given variables X, Y and **known variables**

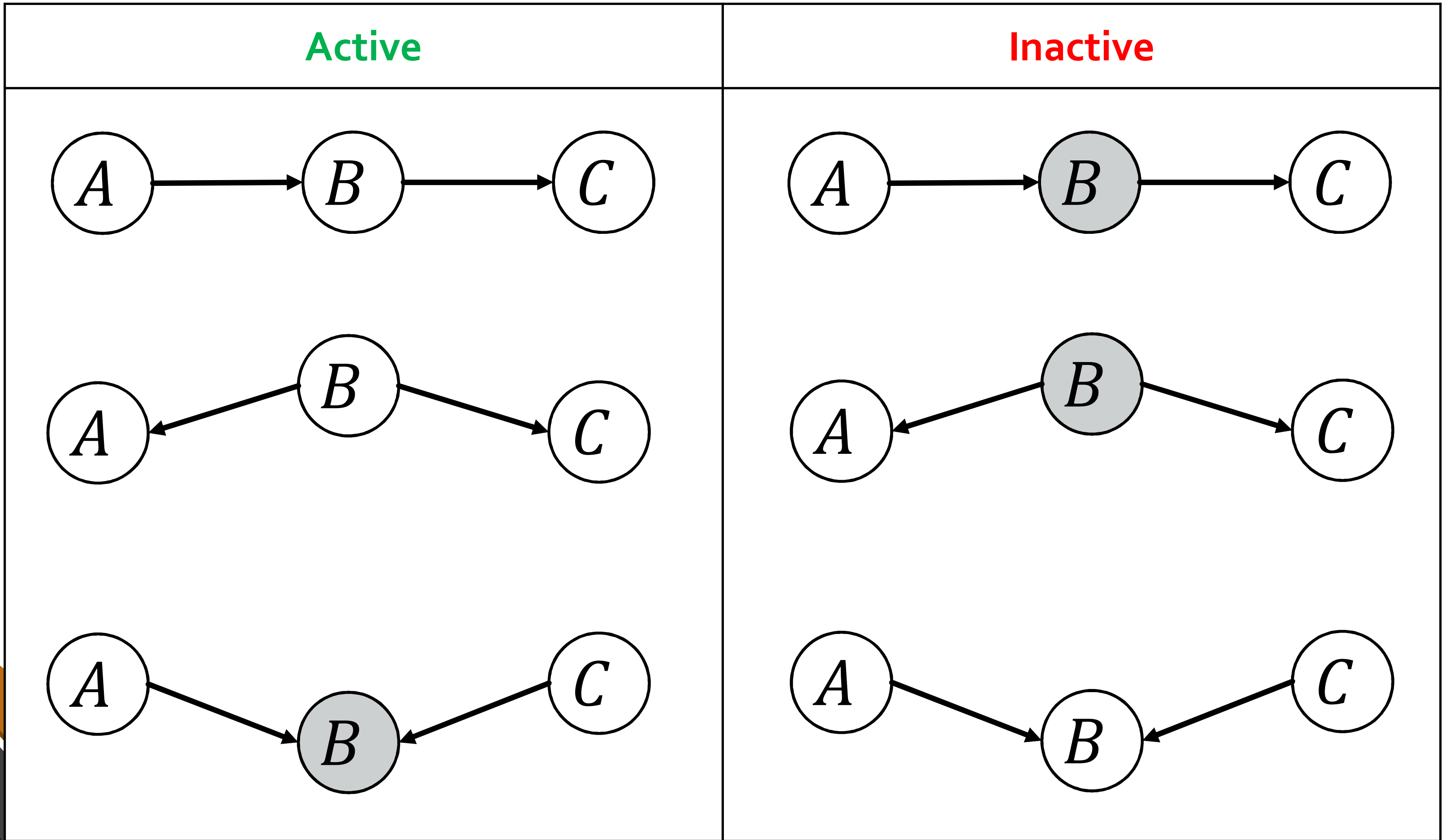
$$\mathcal{E} = \{E_1, \dots, E_k\}$$

are X and Y **surely** independent given $\mathcal{E}$?

- Check every **undirected** path between X and Y (ignore direction of arcs).
- If all paths are not **active** then X and Y are independent given $\mathcal{E}$.

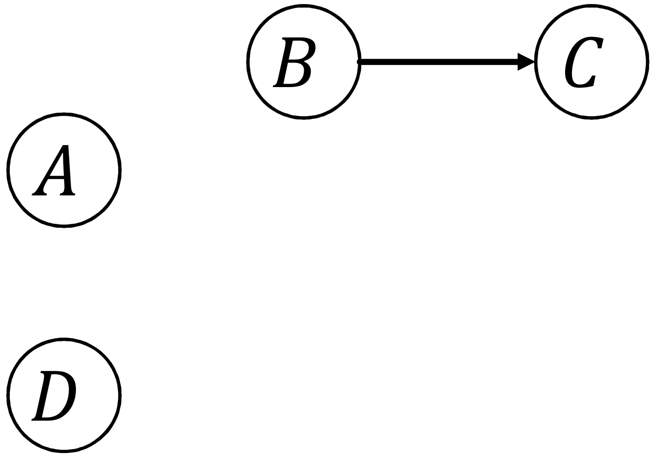
General Case – d Separation

- A path is **active** iff every triple on path is active
- **One** inactive triple ⇒ path is **inactive**!



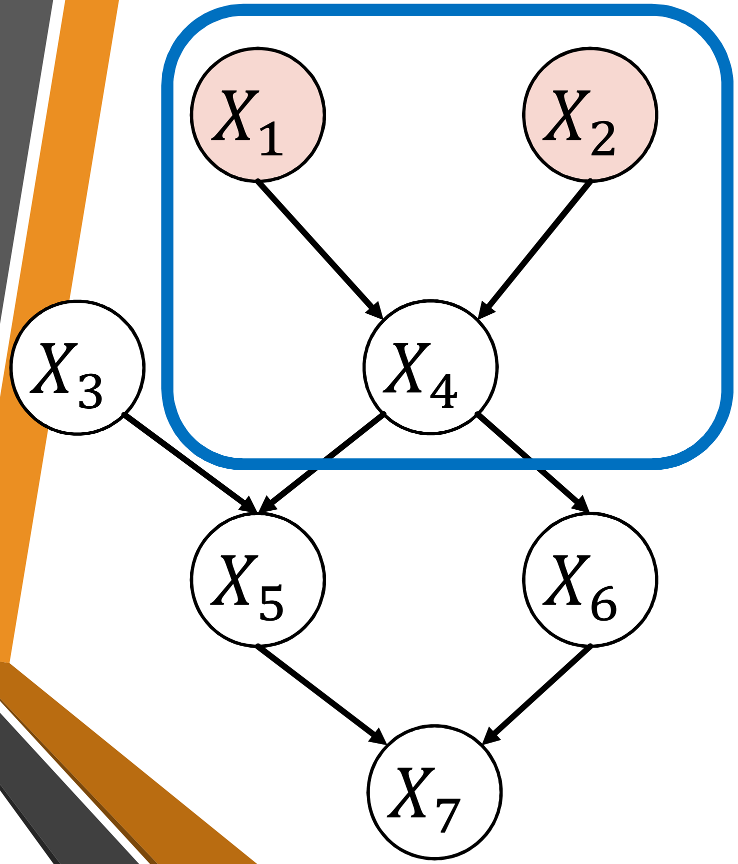
General Case – **Analyze the Graph**

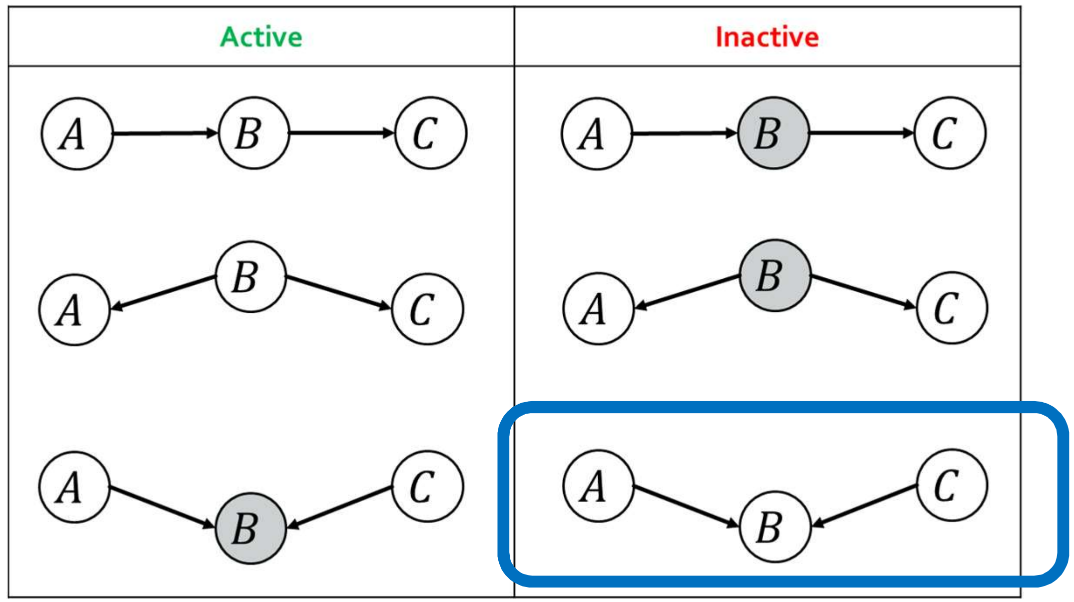
- Degenerate cases:
  - Disconnected variables: always independent.
  - Directly connected variables: never **surely** independent.



General Case – Analyze the Graph

- All paths must be inactive.
- A path is **active** iff every triple on path is active
- **One** inactive triple ⇒ path is **inactive**!



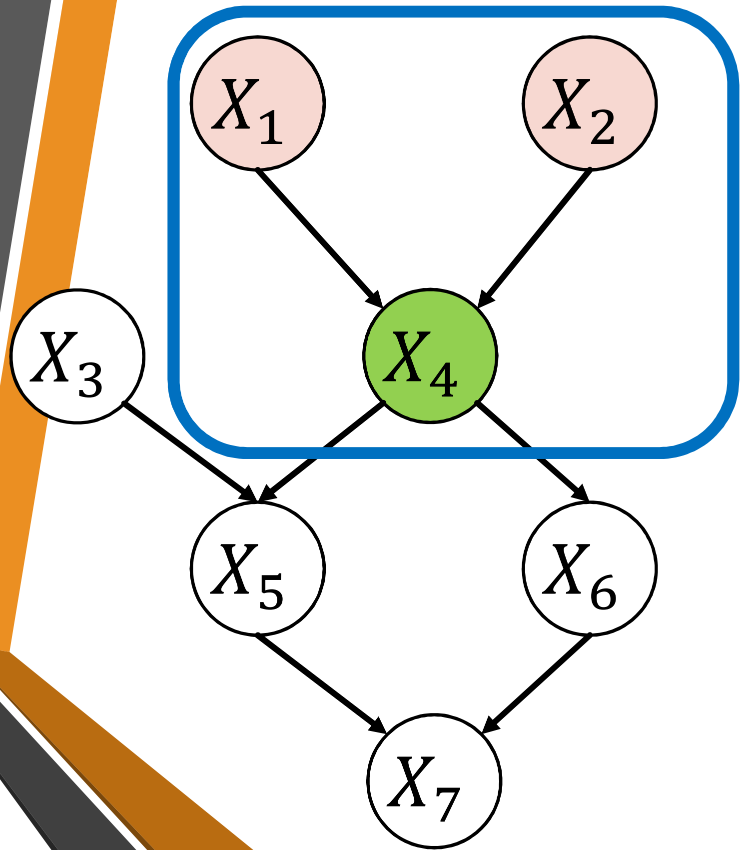


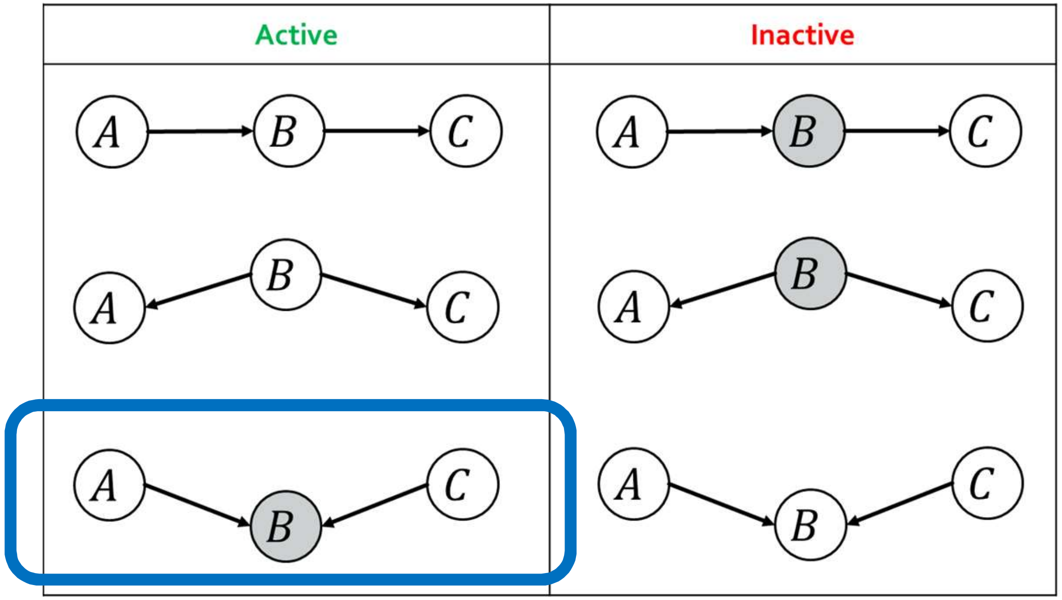
Is X_1 independent of X_2?

Yes!

General Case – Analyze the Graph

- All paths must be inactive.
- A path is **active** iff every triple on path is active
- **One** inactive triple ⇒ path is **inactive**!



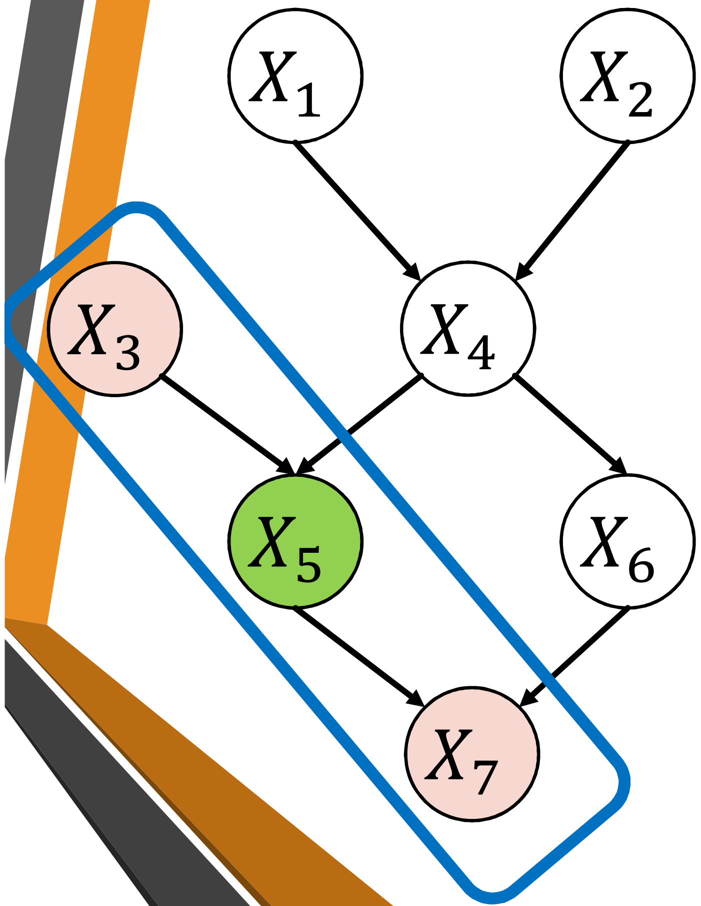


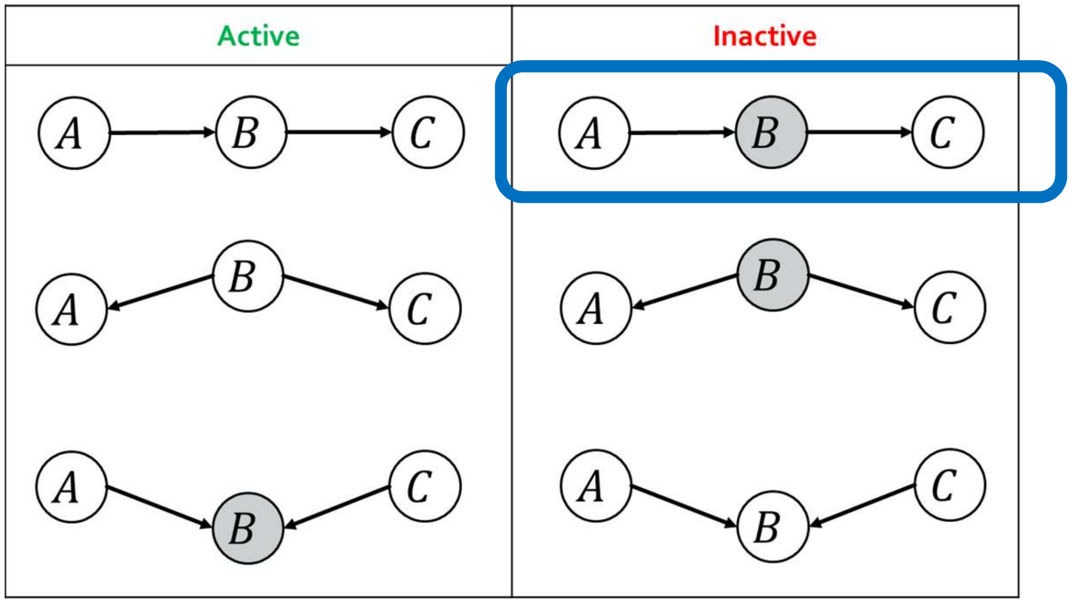
Is X_1 independent of X_2 given X_4?

No!

General Case – Analyze the Graph

- All paths must be inactive.
- A path is **active** iff every triple on path is active
- **One** inactive triple ⇒ path is **inactive**!

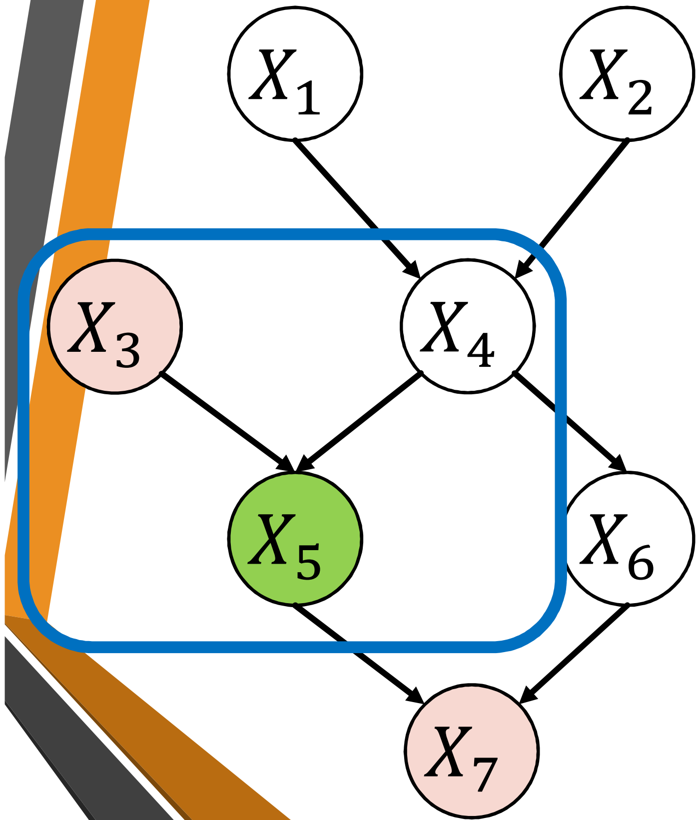


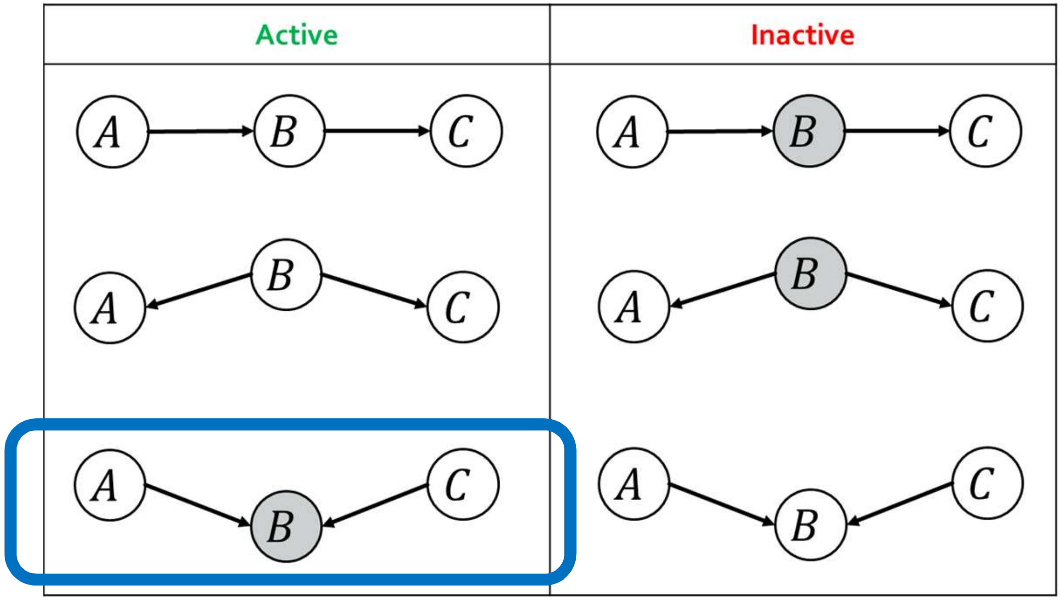


Is X_3 independent of X_7 given X_5?

General Case – Analyze the Graph

- All paths must be inactive.
- A path is **active** iff every triple on path is active
- **One** inactive triple ⇒ path is **inactive**!

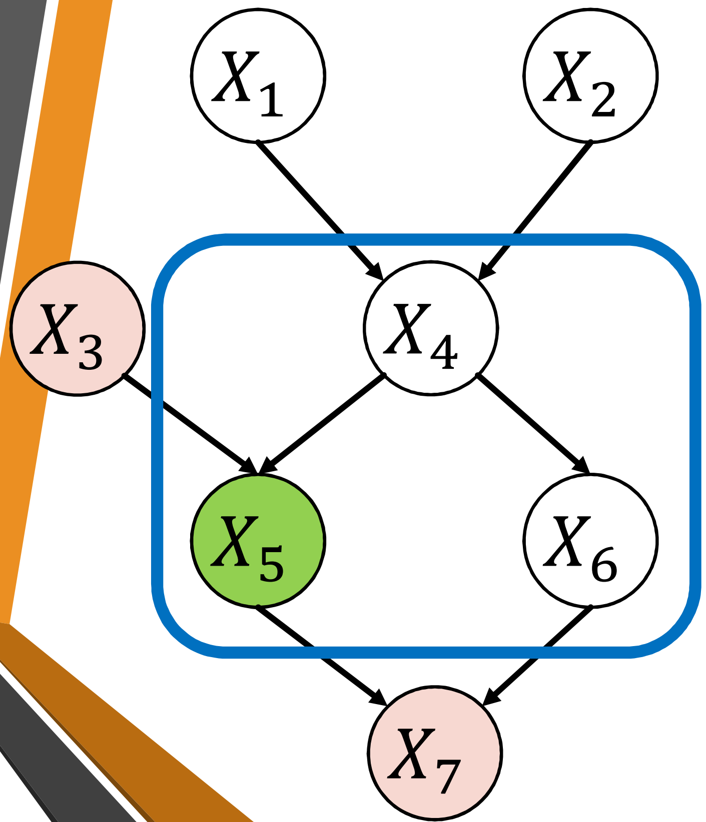


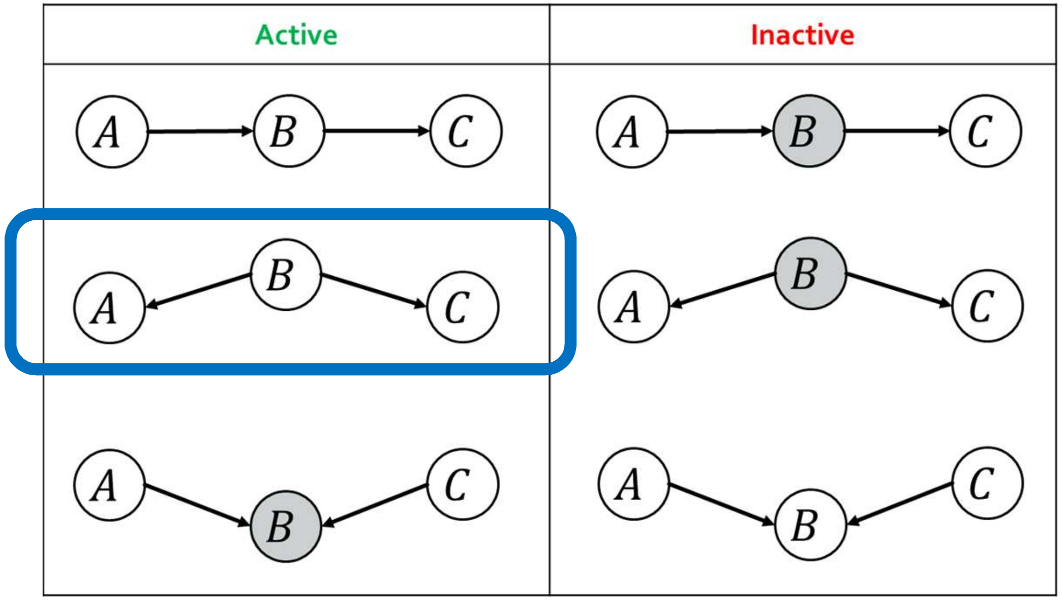


Is X_3 independent of X_7 given X_5?

General Case – Analyze the Graph

- All paths must be inactive.
- A path is **active** iff every triple on path is active
- **One** inactive triple ⇒ path is **inactive**!

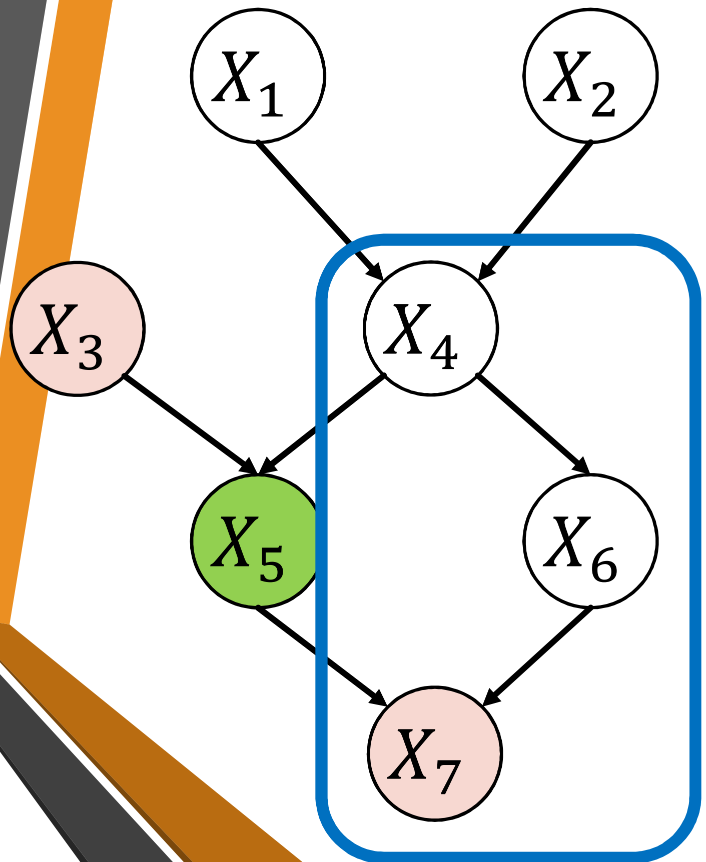


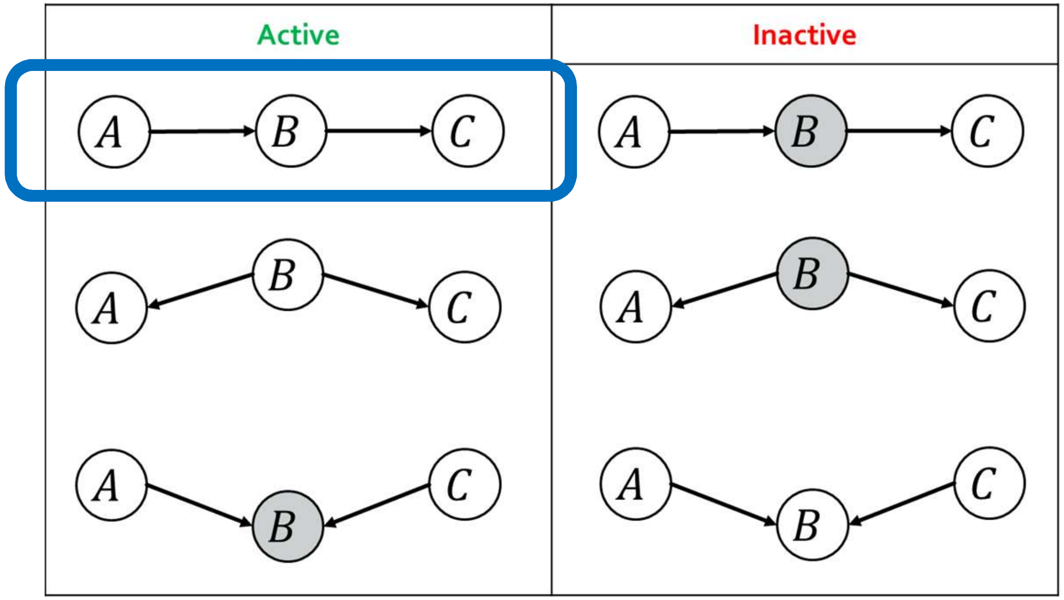


Is X_3 independent of X_7 given X_5?

General Case – Analyze the Graph

- All paths must be inactive.
- A path is **active** iff every triple on path is active
- **One** inactive triple ⇒ path is **inactive**!



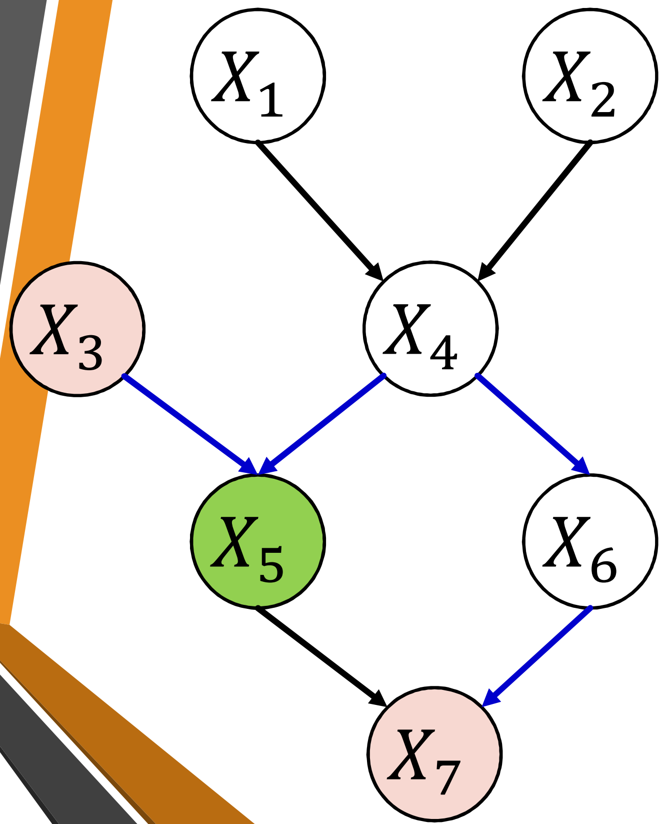


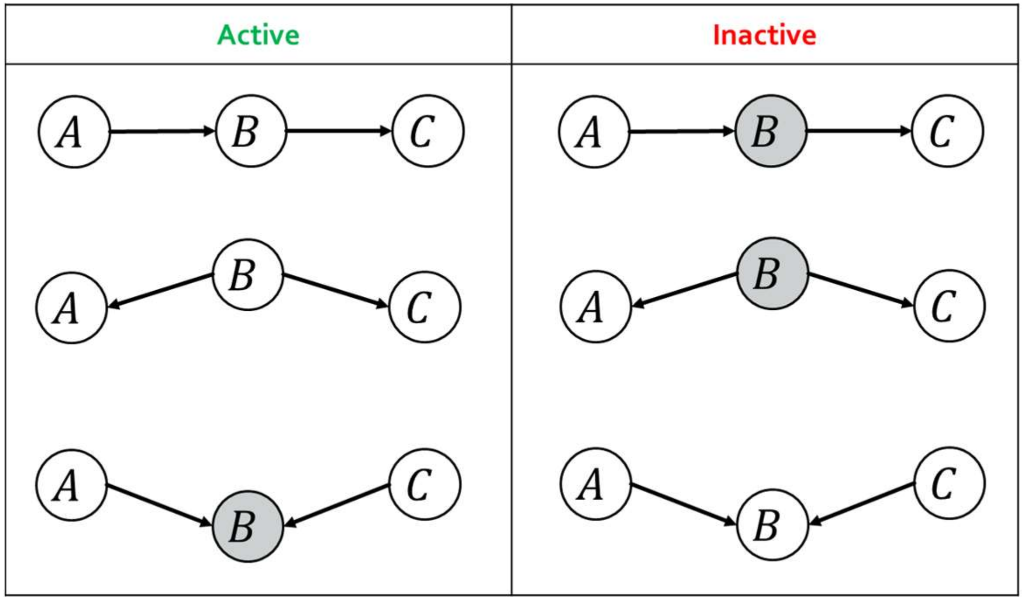
Is X_3 independent of X_7 given X_5?

No!

General Case – Analyze the Graph

- All paths must be inactive.
- A path is **active** iff every triple on path is active
- **One** inactive triple ⇒ path is **inactive**!

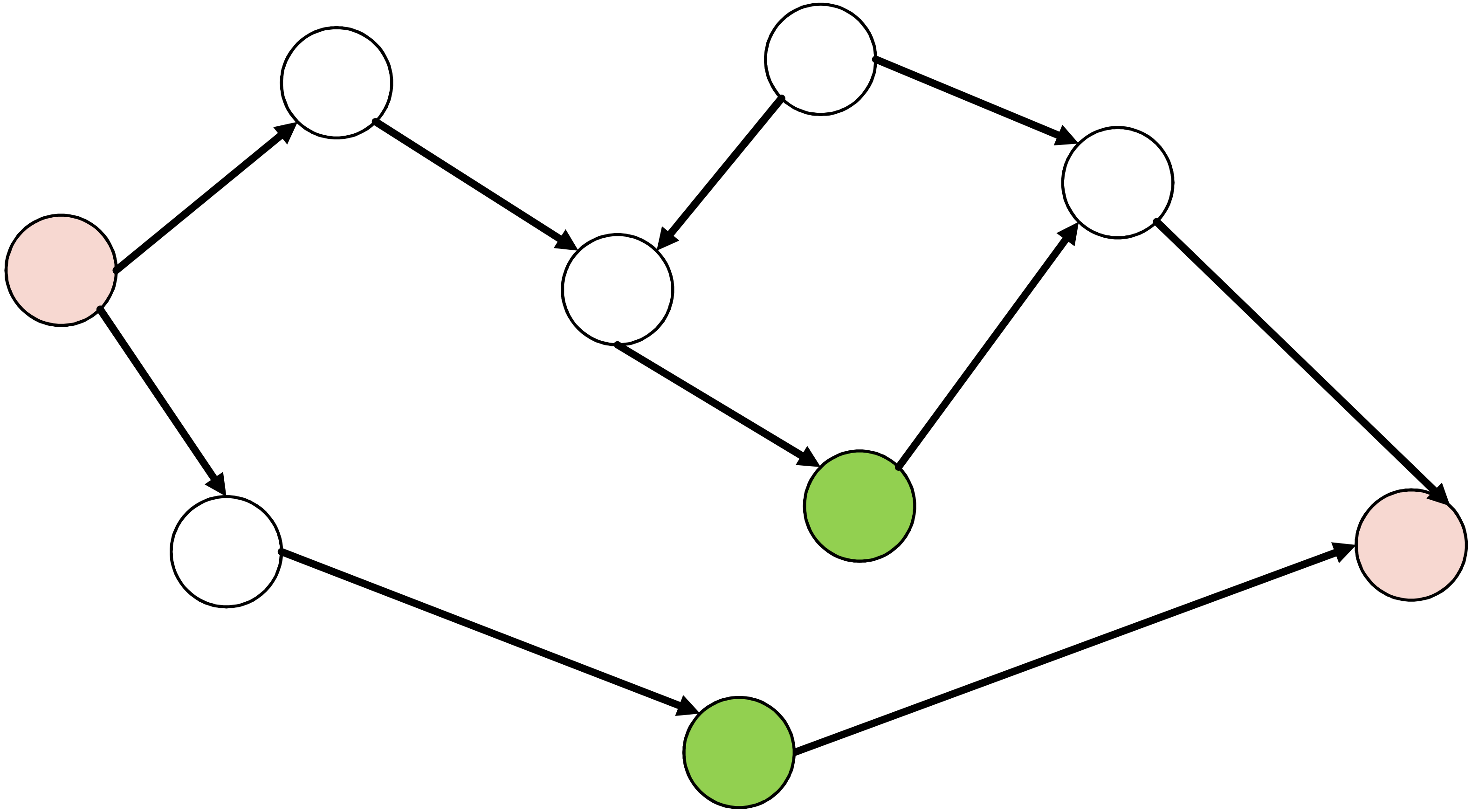




Is X_3 independent of X_7 given X_5?

No!

X_3, X_5, X_4, X_6, X_7 form an **active path**



Learning From Examples

AIMA Chapter 18

Learning – What and Why?

- An agent is said to be ***learning*** if it improves its performance P on task T based on experience/observations/data E
  - T must be fixed, P must be measurable, E must exist
  - e.g., image recognition, game playing, driving in urban environments…

Reasons for learning:

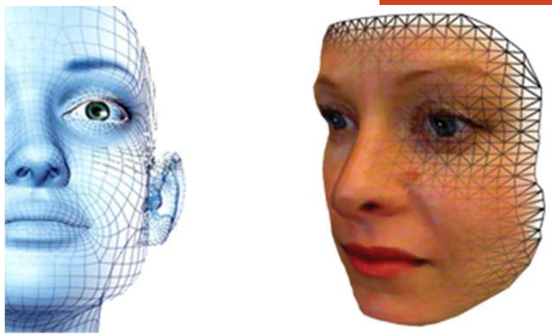
- Unknown, dynamically changing task environments
- Hard to encode all knowledge (e.g., face recognition)
- Easier to program

ML is Everywhere!

Natural Language Processing

Computational Biology

Targeted Advertising



Object Recognition

Designing Learning Elements

Design of learning elements

- What are we learning?
- Data represented?
- Performance feedback?

Feedback

- **Supervised learning:** each example is labeled
- **Unsupervised learning:** correct answers not given
- **Reinforcement learning:** occasional rewards given

Supervised Learning

Key idea: Learn an unknown function f from examples

The Data:

- We are given a dataset $\mathcal{X} \subseteq \mathbb{R}^n$
- Each datapoint $\vec{x} \in \mathcal{X}$ has a label $f(\vec{x})$ (for now, $f(\vec{x}) \in \{\pm 1\}$).

Problem

- Search for hypothesis $h \in \mathcal{H}$ such that $h \simeq f$.

Performance:

- Measured over examples that are distinct from training set
- Hypothesis generalizes well if it correctly predicts the value of f for novel examples.

Measuring Error

Where does the data come from? Distribution $\mathcal{D}$

Key idea: Learn an unknown function f from examples

- Given a hypothesis $h \in \mathcal{H}$, let the loss wrt $\mathcal{D}$
$$L_{\mathcal{D}}(h) = \Pr_{\vec{x} \sim \mathcal{D}}[h(\vec{x}) \neq f(\vec{x})]$$
- The error of $\mathcal{H}$: the best that we can do with $\mathcal{H}$
$$OPT_{\mathcal{D}}(\mathcal{H}) = \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h)$$
- Nearly optimal error:
$$L_{\mathcal{D}}(h) \leq OPT_{\mathcal{D}}(\mathcal{H}) + \varepsilon$$
- Given a set of samples $\mathcal{X} = \{\vec{x}_1, \dots, \vec{x}_m\}$, let the empirical error be:
$$\hat{L}_{\mathcal{X}}(h) = \frac{1}{m} \sum_{j=1}^{m} \mathbb{I}\left(h(\vec{x}_j) \neq f(\vec{x}_j)\right)$$

What is the probability that h will misclassify

Probably Approximately Correct Learning

Key idea: Learn an unknown function f from examples

Efficient PAC Learning

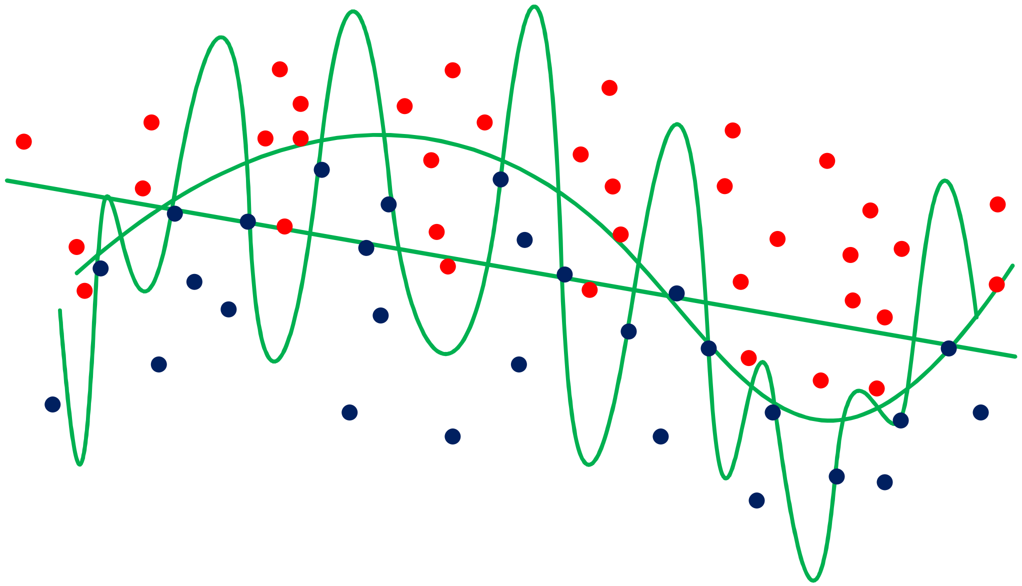
Approximately correct part of PAC

- The hypothesis is likely to match f on future samples
  $L_{\mathcal{D}}(h) = \Pr_{\vec{x} \sim \mathcal{D}}[h(\vec{x}) \neq f(\vec{x})] < OPT_{\mathcal{D}}(\mathcal{H}) + \varepsilon$
- Guarantee holds with probability $\geq 1 - \delta$
- **Sample Complexity:** $|\mathcal{X}|$ polynomial in $n, \frac{1}{\varepsilon}, \log\frac{1}{\delta}$

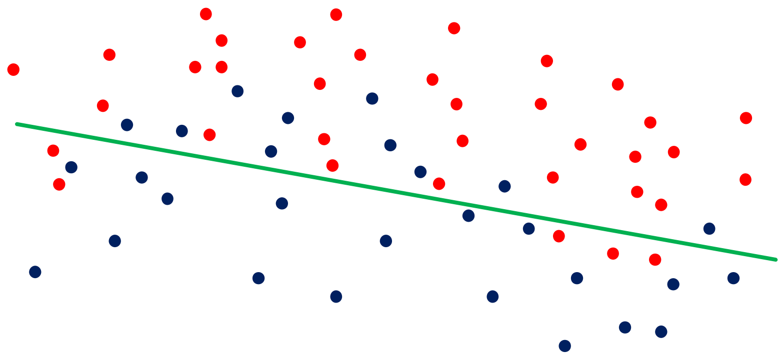
Probably part of PAC.
How likely is it that the sample from $\mathcal{D}$ when drawn IID is bad?
Should be very small!

Learning a Good Classifier

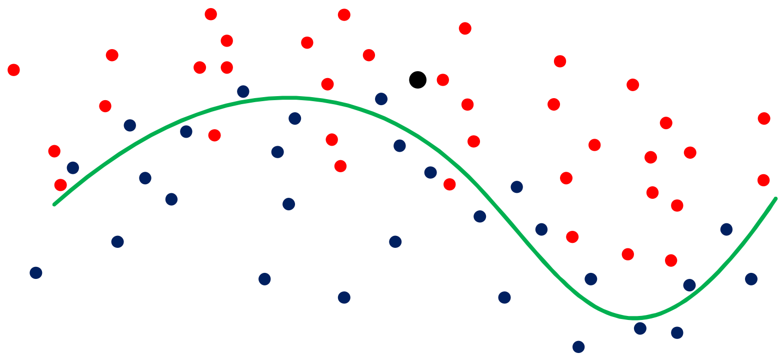
- Our goal is to find a function $h: \mathbb{R}^n \to \{-1,1\}$ that fits the data: *what is the likeliest function to have generated our dataset?*



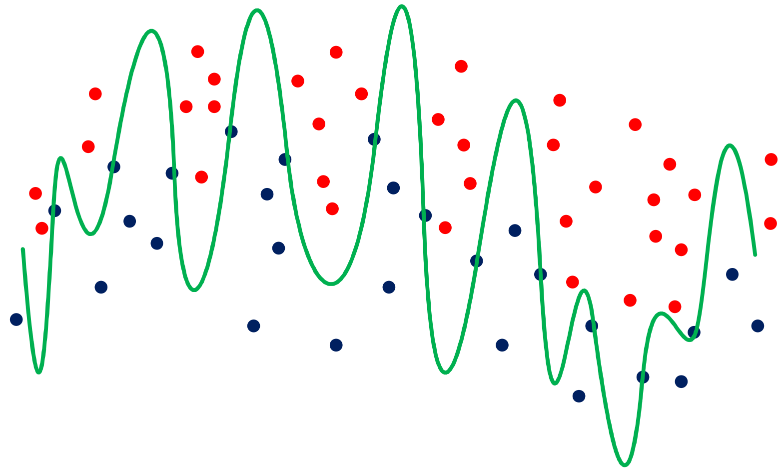
I.



II



III.



Simple Explanations
– “Occam’s Razor”
vs.
Low error rate

Choosing Simple Hypotheses

- Generalize better in the presence of noisy data
- Faster to search through simple hypothesis space
- Easier and faster to use simple hypothesis

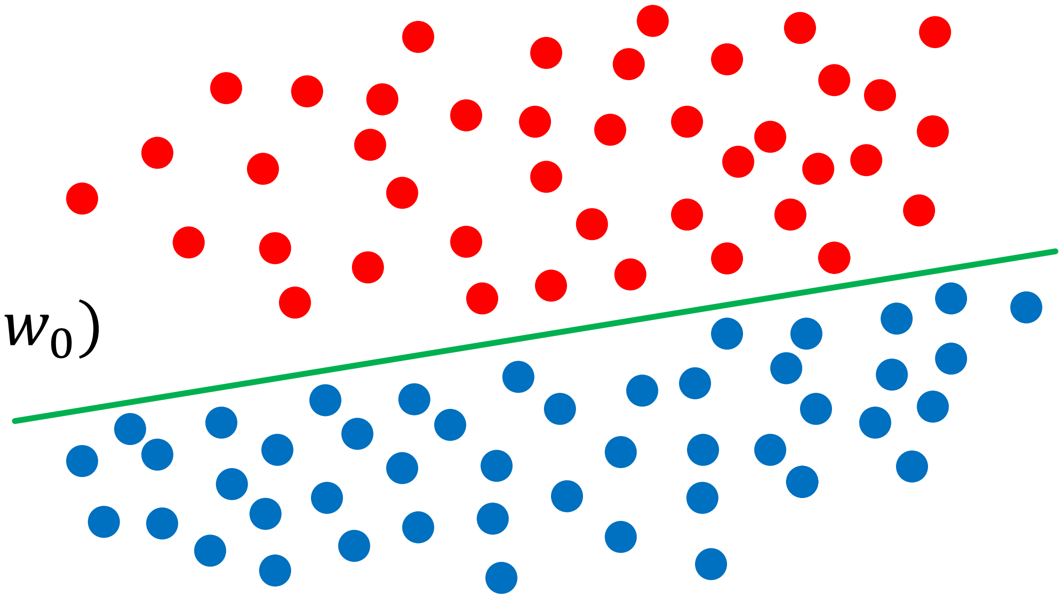
Linear Classifiers

Assumption: data was generated by some linear function

$$f(\vec{x}) = \begin{cases} 1 \text{ if } \vec{w}^T\vec{x} - w_0 \geq 0 \\ -1 \text{ otherwise.} \end{cases}$$

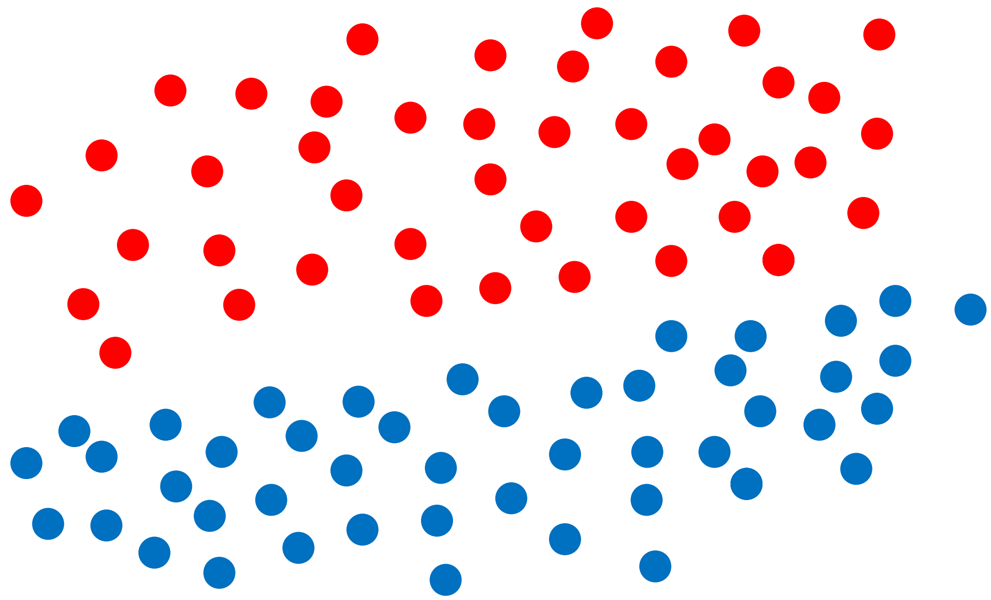
f is called a *linear classifier*.

$$f(x, y) = \mathbb{I}(w_1 x + w_2 y \geq w_0)$$



Linear Classifiers

Question: given a dataset, how would we determine which linear classifier is "good"?



Least Squared Error

Let $y(\vec{w}, w_0, \vec{x}) = \mathbb{I}(\vec{w}^T \vec{x} \geq w_0)$

Empirical error:

given a data point $\vec{x}_j$ labelled t_j, our classifier is wrong if

$$y(\vec{w}, w_0, \vec{x}_j) \neq t_j$$

Objective – minimize empirical loss:

$$\min_{\vec{w} \in \mathbb{R}^n, w_0} \sum_j \left(y(\vec{w}, w_0, \vec{x}_j) - t_j\right)^2$$

"find $\vec{w}, w_0$ that minimize the total number of mistakes on dataset"

Other loss functions are possible!

Regularization

$$\min_{\vec{w},w_0} \underbrace{\frac{1}{m}\sum_{j=1}^{m}\left(y(\vec{w},w_0,\vec{x}_j)-t_j\right)^2}_{\text{Empirical Loss}} + \underbrace{\lambda\|\vec{w}\|}_{\text{Regularization}}$$

Minimize: Empirical Loss $\hat{L}(h)$ + Regularization $R(h)$

- If there is some hypothesis $h \in \mathcal{H}$ for which $\hat{L}(h) = 0$, then for a sufficiently small $\lambda > 0$, we are just minimizing $\|\vec{w}\|$.
- In general: tradeoff between **simplicity** and **accuracy**.

CS3243: Introduction to AI

Concluding Remarks

Assessment: Reminder

What	When	Grade Percentage
~~Midterm Exam~~	~~**30 September 2019**~~	~~20%~~
Final Exam	**26 November 2019** (morning)	40%
~~2 Assignments~~	~~As announced!~~	~~20%~~
3rd Assignments	As announced!	10%
Participation (lecture + tutorial)	-	10%

Final Exam - **26 November 2019** (morning)

Where? – Check on Edurec (MPSH/UT)

What Can You Bring

- Closed book assessment.
- 1 A4-sized double-sided sheet of personal notes. (No appendages!)
- **NUS APPROVED CALCULATOR**

Structure

- 2 parts – MCQ and structured questions
- **Bring 2B pencil**
- Writing in pencil is allowed

Topics for Final Exam

Topics to be covered

- Chapter 2 – Agents
- Chapter 3.1 to 3.4 – Uninformed Search
- Chapter 3.5.1 to 3.5.2, 3.6.1 to 3.6.2 – Informed Search
- Chapter 4.1 – Local Search
- Chapter 5.1 to 5.5 – Adversarial Search
- Chapter 6.1 to 6.4 – Constraint Satisfaction Problems
- Chapter 7 – Logical Agents, Propositional Logic & Inference
- Chapter 8, 9.1 to 9.3, 9.4.1, 9.5.1 to 9.5.3 – First-Order Logic & Inference
- Chapter 13 – Uncertainty
- Chapter 14.1 to 14.2 – Bayesian Networks

Summary: What Have You Learned?

First half of CS3243:

- Assume complete environment knowledge.
- Make decisions, search for solutions

Second half of CS3243:

- Incomplete information
- Gather information/knowledge, infer/predict environment data, to make better decisions

Key Idea #1 – Search

Basic Model Paradigm

- Informed
- Uninformed
- Adversarial

CSPs + Logic

- Finding solutions as search problem

Optimization

- Optimal solutions found via search

Search cutoff strategies

- Heuristic functions
- Heuristics in CSPs/Logic

Key Idea #2 - Randomization

Search

- makes it faster, avoids getting stuck in bad local optimal states.

Heuristics

- needs to be corrected

Strategic play

- abstraction and expected utility

In SAT solving

- obtain nearly optimal solutions quickly

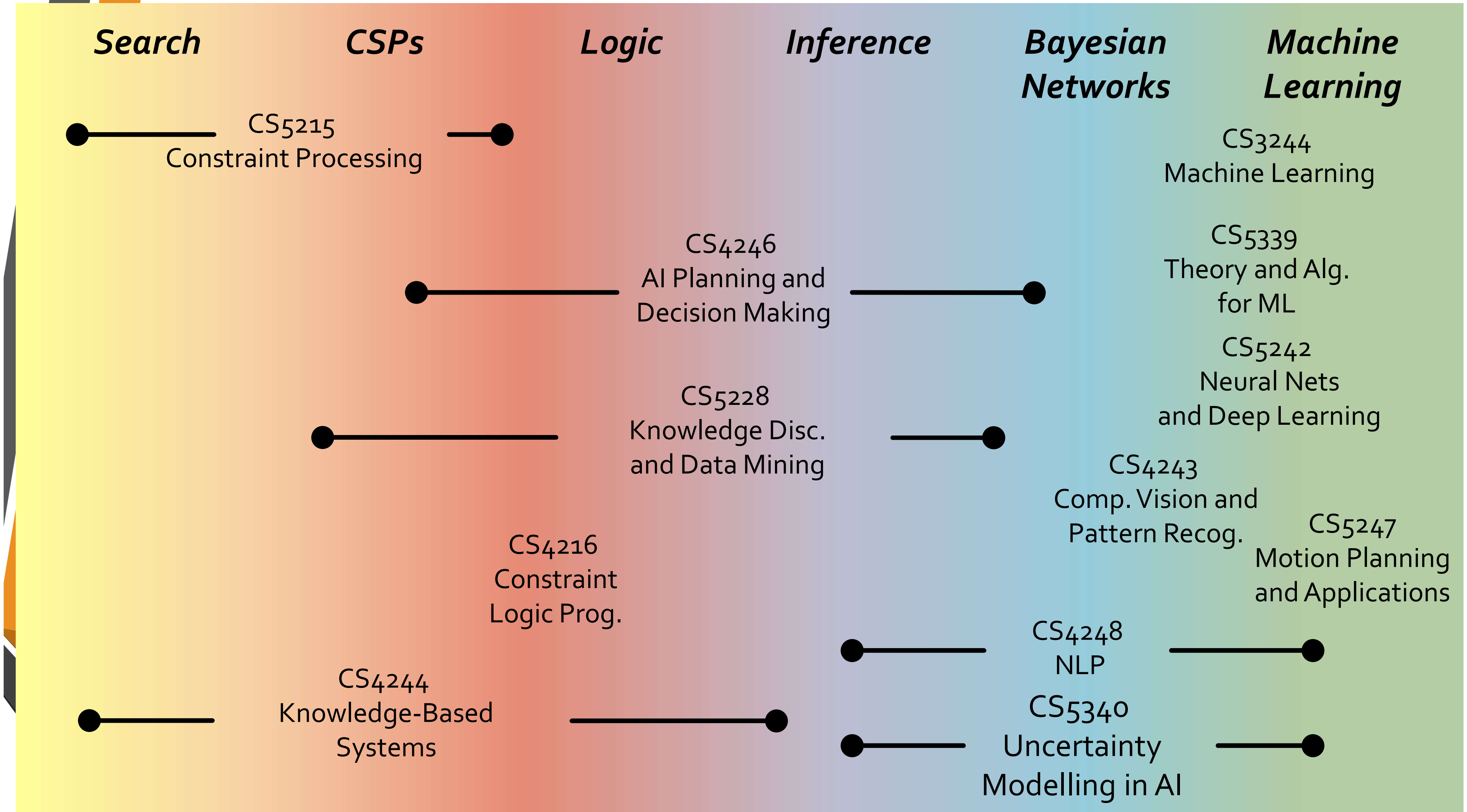
Bayesian inference

- basic building block of inference under uncertainty

Learning

- approximately optimal learners from randomly generated datasets.

From Appreciating to Applying AI



Any Final Questions?

Keep in touch: anarayan@comp.nus.edu.sg