



10-301/10-601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Bayesian Networks + Reinforcement Learning: Markov Decision Processes

Matt Gormley
Lecture 21
Apr. 3, 2023

Reminders

- **Homework 7: Hidden Markov Models**
 - **Out: Fri, Mar. 31**
 - **Due: Mon, Apr. 10 at 11:59pm**

**GRAPHICAL MODELS:
DETERMINING CONDITIONAL
INDEPENDENCIES**

What Independencies does a Bayes Net Model?

- In order for a Bayesian network to model a probability distribution, the following must be true:

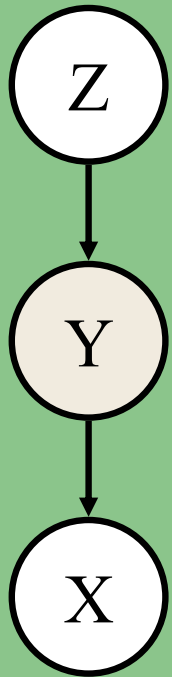
Each variable is conditionally independent of all its non-descendants in the graph given the value of all its parents.

- This follows from
$$P(X_1, \dots, X_T) = \prod_{t=1}^T P(X_t \mid \text{parents}(X_t))$$
$$= \prod_{t=1}^T P(X_t \mid X_1, \dots, X_{t-1})$$
- But what else does it imply?

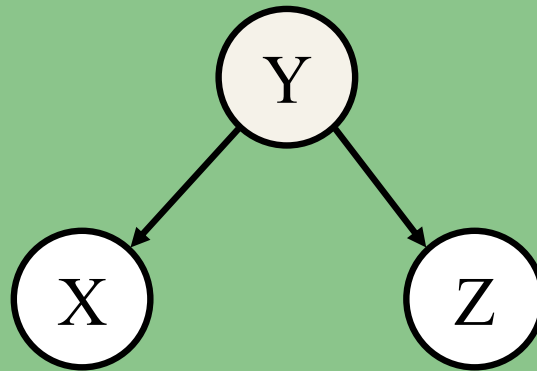
What Independencies does a Bayes Net Model?

Three cases of interest...

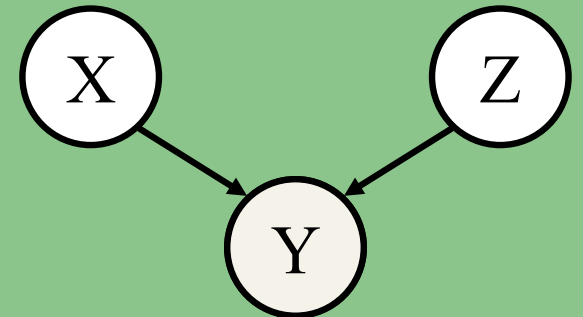
Cascade



Common Parent



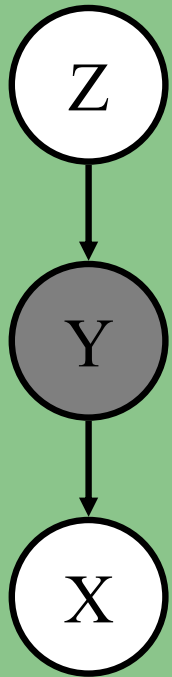
V-Structure



What Independencies does a Bayes Net Model?

Three cases of interest...

Cascade ✓

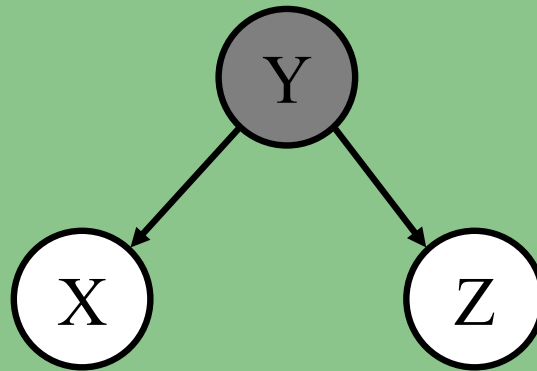


$$X \perp\!\!\!\perp Z \mid Y$$

Knowing Y
decouples X and Z

Common Parent

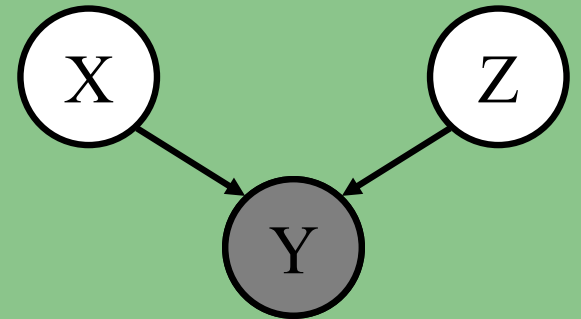
similar proof to cascade



$$X \perp\!\!\!\perp Z \mid Y$$

V-Structure

∃ prob.s that are dependent

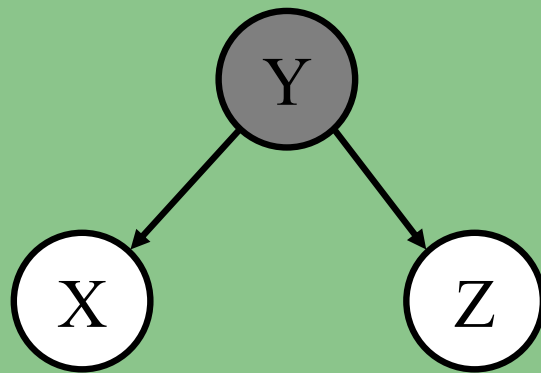


$$X \not\perp\!\!\!\perp Z \mid Y$$

Knowing Y
couples X and Z

Whiteboard

Common Parent



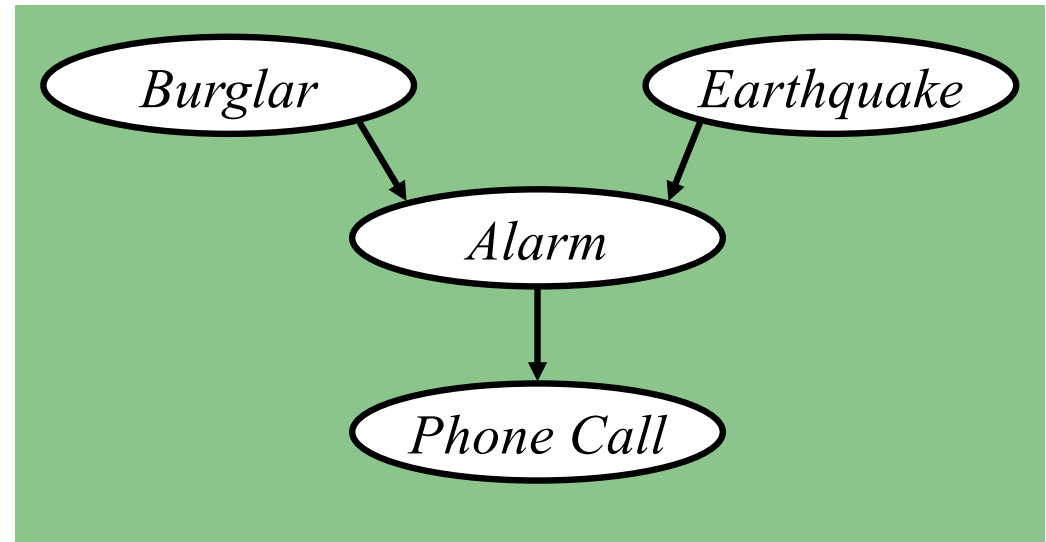
Proof of
conditional
independence

(The other two
cases can be
shown just as
easily.)

$$X \perp\!\!\!\perp Z \mid Y$$

The “Burglar Alarm” example

- Your house has a twitchy burglar alarm that is also sometimes triggered by earthquakes.
- Earth arguably doesn’t care whether your house is currently being burgled
- While you are on vacation, one of your neighbors calls and tells you your home’s burglar alarm is ringing. Uh oh!



Quiz: True or False?

Q1

A = toxic

B = True

C = False

30%

70%

$Burglar \perp\!\!\!\perp Earthquake \mid PhoneCall$

“explaining away”

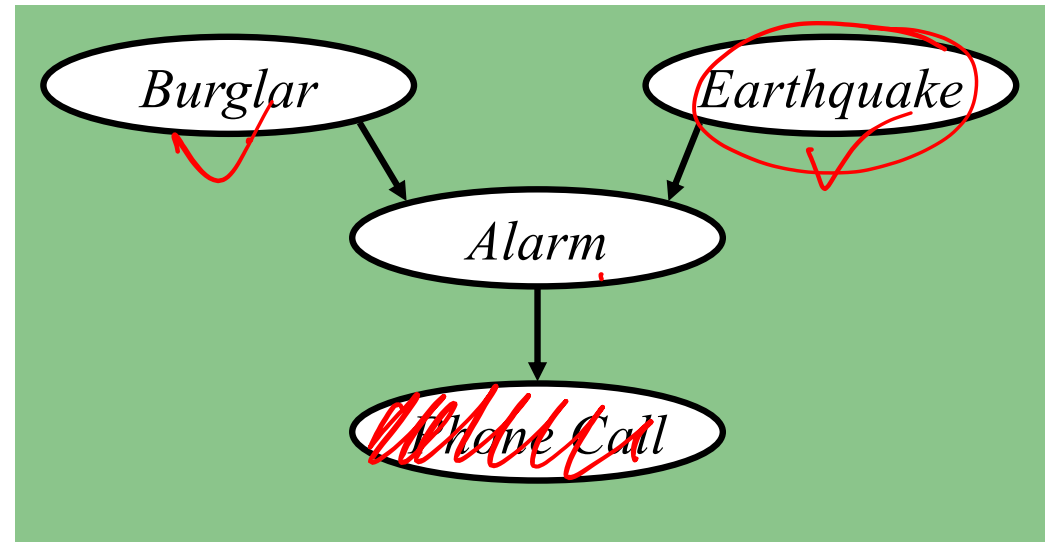
The “Burglar Alarm” example

- After you get this phone call, suppose you learn that there was a medium-sized earthquake in your neighborhood. Oh, whew! Probably not a burglar after all.
- Earthquake “explains away” the hypothetical burglar.
- But then it must **not** be the case that

$$Burglar \perp\!\!\!\perp Earthquake \mid PhoneCall$$

even though

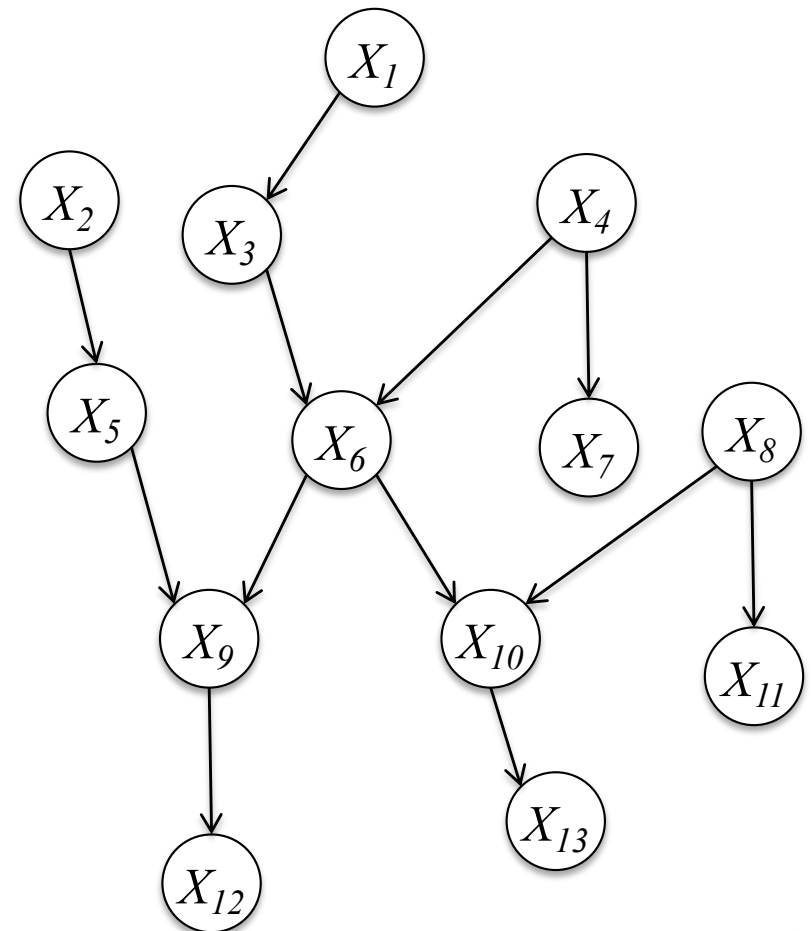
$$Burglar \perp\!\!\!\perp Earthquake$$



Markov Boundary

Def: the **co-parents** of a node are the parents of its children

Def: the **Markov boundary** of a node is the set containing the node's parents, children, and co-parents.

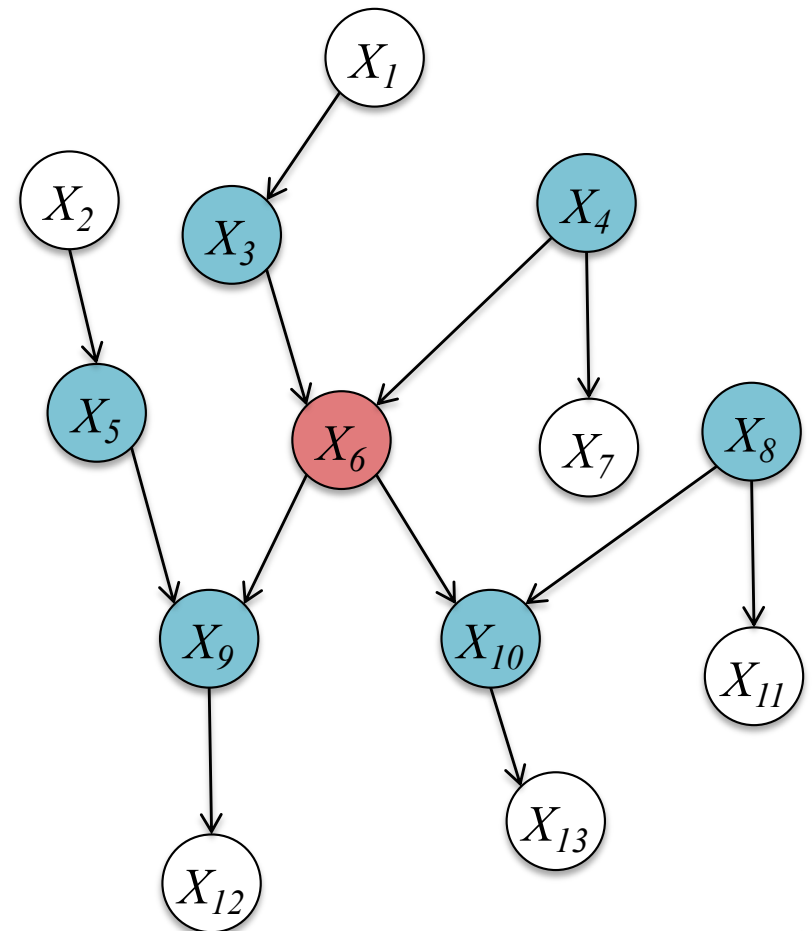


Markov Boundary

Def: the **co-parents** of a node are the parents of its children

Def: the **Markov boundary** of a node is the set containing the node's parents, children, and co-parents.

Example: The Markov boundary of X_6 is $\{X_3, X_4, X_5, X_8, X_9, X_{10}\}$



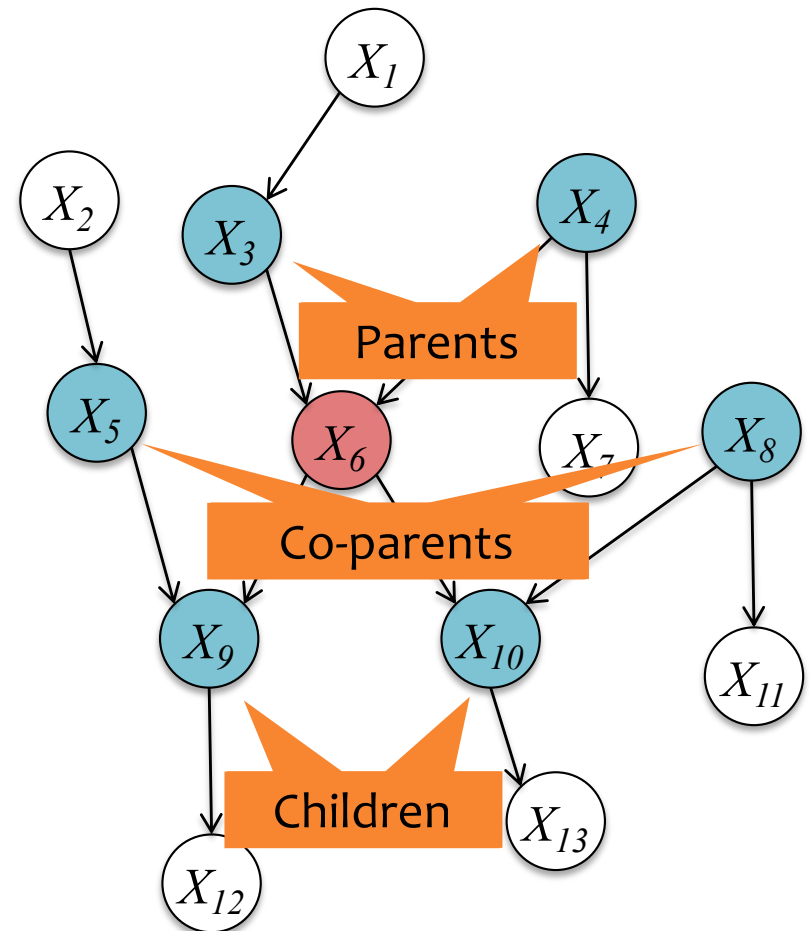
Markov Boundary

Def: the **co-parents** of a node are the parents of its children

Def: the **Markov boundary** of a node is the set containing the node's parents, children, and co-parents.

Theorem: a node is **conditionally independent** of every other node in the graph given its **Markov boundary**

Example: The Markov boundary of X_6 is $\{X_3, X_4, X_5, X_8, X_9, X_{10}\}$



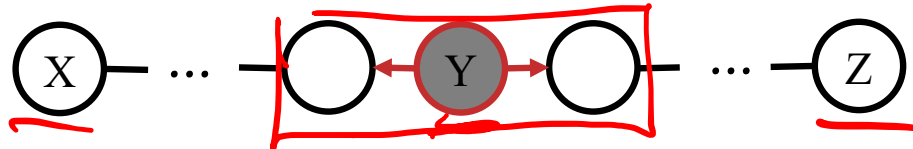
D-Separation

Definition #1:

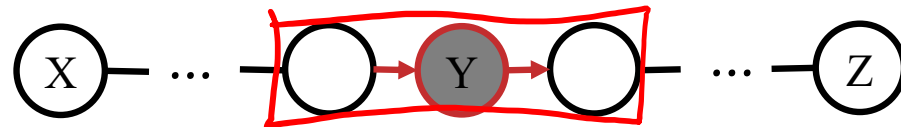
Variables X and Z are **d-separated** given a **set** of evidence variables E (variables that are observed) iff every path from X to Z is “blocked”.

A path is “blocked” whenever:

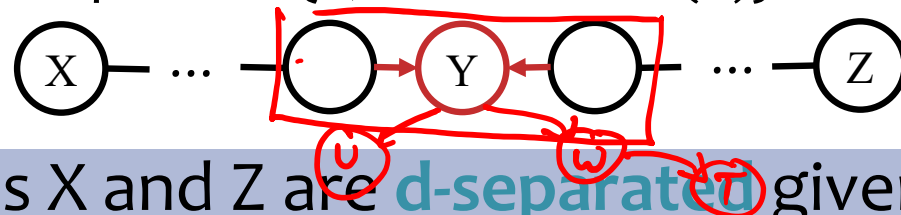
1. $\exists Y$ on path s.t. $Y \in E$ and Y is a “common parent”



2. $\exists Y$ on path s.t. $Y \in E$ and Y is in a “cascade”



3. $\exists Y$ on path s.t. $\{Y, \text{descendants}(Y)\} \notin E$ and Y is in a “v-structure”



If variables X and Z are **d-separated** given a **set** of variables E
Then X and Z are **conditionally independent** given the **set** E

D-Separation

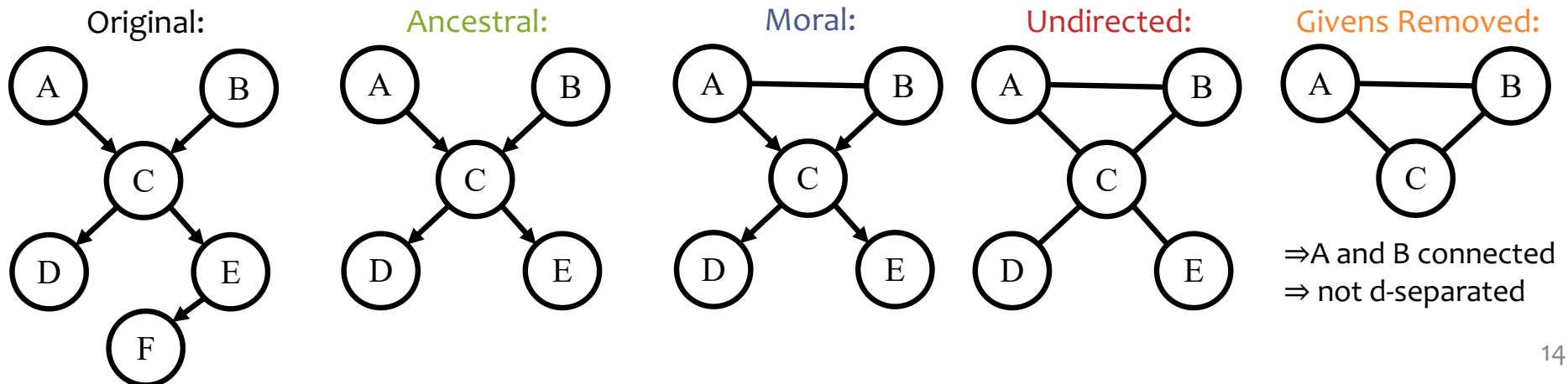
If variables X and Z are **d-separated** given a **set** of variables E
Then X and Z are **conditionally independent** given the **set** E

Definition #2:

Variables X and Z are **d-separated** given a **set** of evidence variables E iff there does **not** exist a path between X and Z in the **undirected ancestral moral** graph **with E removed**.

1. **Ancestral graph:** keep only X, Z, E and their ancestors
2. **Moral graph:** add undirected edge between all pairs of each node's parents
3. **Undirected graph:** convert all directed edges to undirected
4. **Givens Removed:** delete any nodes in E

Example Query: $A \perp\!\!\!\perp B \mid \{D, E\}$



SUPERVISED LEARNING FOR BAYES NETS

Machine Learning

The **data** inspires the structures we want to predict

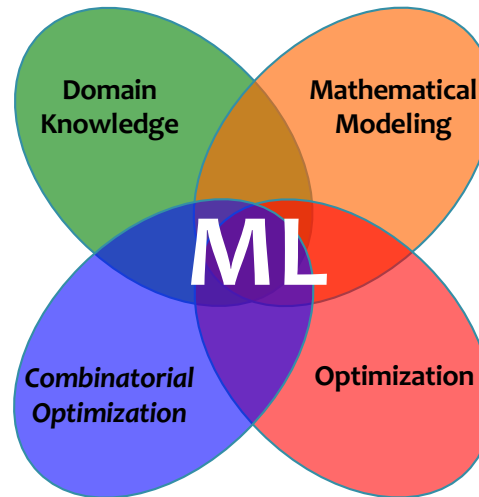


Our **model** defines a score for each structure

It also tells us what to optimize



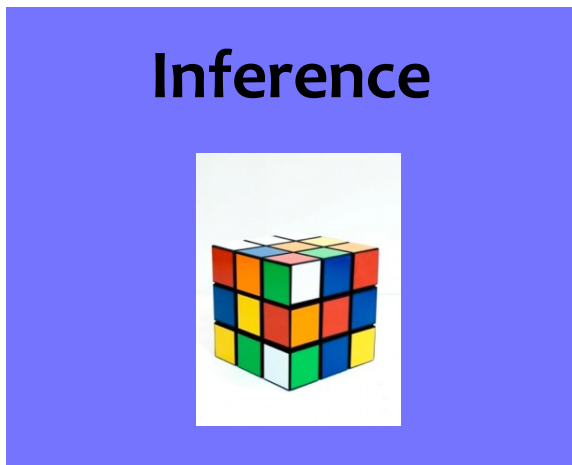
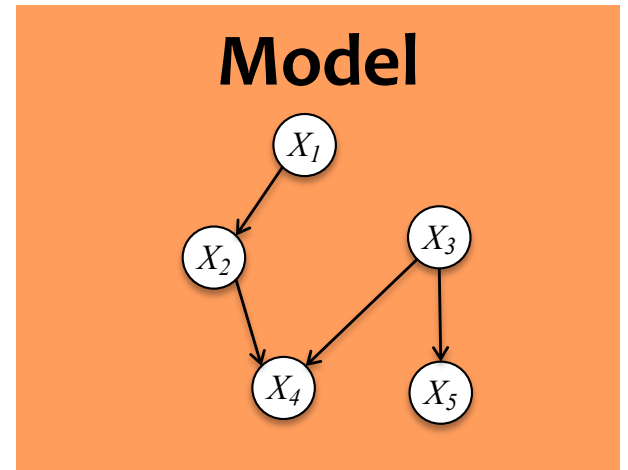
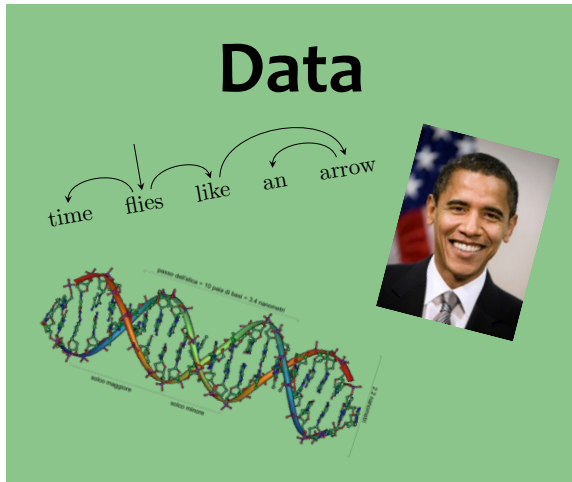
Learning tunes the parameters of the model



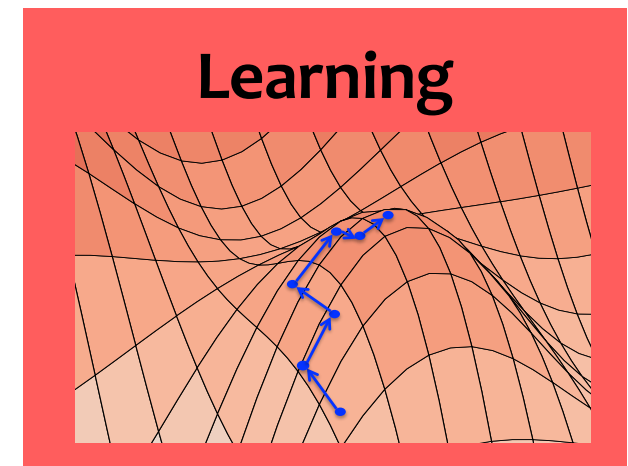
Inference finds {best structure, marginals, partition function} for a new observation

(**Inference** is usually called as a subroutine in learning)

Machine Learning



(Inference is usually called as a subroutine in learning)



Recipe for Closed-form MLE

1. Assume data was generated i.i.d. from some model (i.e. write the generative story)

$$x^{(i)} \sim p(x|\boldsymbol{\theta})$$

2. Write log-likelihood

$$\ell(\boldsymbol{\theta}) = \log p(x^{(1)}|\boldsymbol{\theta}) + \dots + \log p(x^{(N)}|\boldsymbol{\theta})$$

3. Compute partial derivatives (i.e. gradient)

$$\partial \ell(\boldsymbol{\theta}) / \partial \theta_1 = \dots$$

$$\partial \ell(\boldsymbol{\theta}) / \partial \theta_2 = \dots$$

...

$$\partial \ell(\boldsymbol{\theta}) / \partial \theta_M = \dots$$

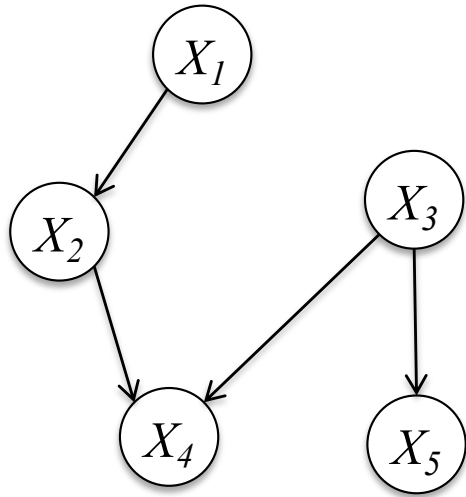
4. Set derivatives to zero and solve for $\boldsymbol{\theta}$

$$\partial \ell(\boldsymbol{\theta}) / \partial \theta_m = 0 \text{ for all } m \in \{1, \dots, M\}$$

$$\boldsymbol{\theta}^{\text{MLE}} = \text{solution to system of } M \text{ equations and } M \text{ variables}$$

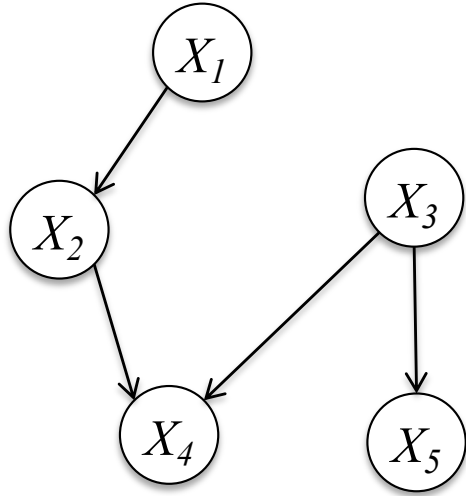
5. Compute the second derivative and check that $\ell(\boldsymbol{\theta})$ is concave down at $\boldsymbol{\theta}^{\text{MLE}}$

Learning Fully Observed BNs



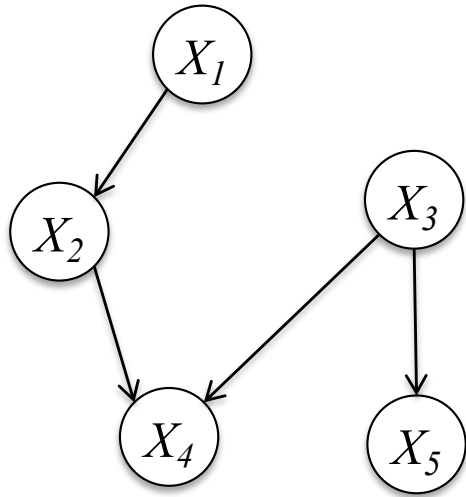
$$\begin{aligned} p(X_1, X_2, X_3, X_4, X_5) = & \\ & p(X_5 | X_3) p(X_4 | X_2, X_3) \\ & p(X_3) p(X_2 | X_1) p(X_1) \end{aligned}$$

Learning Fully Observed BNs



$$p(X_1, X_2, X_3, X_4, X_5) =$$
$$p(X_5|X_3)p(X_4|X_2, X_3)$$
$$p(X_3)p(X_2|X_1)p(X_1)$$

Learning Fully Observed BNs

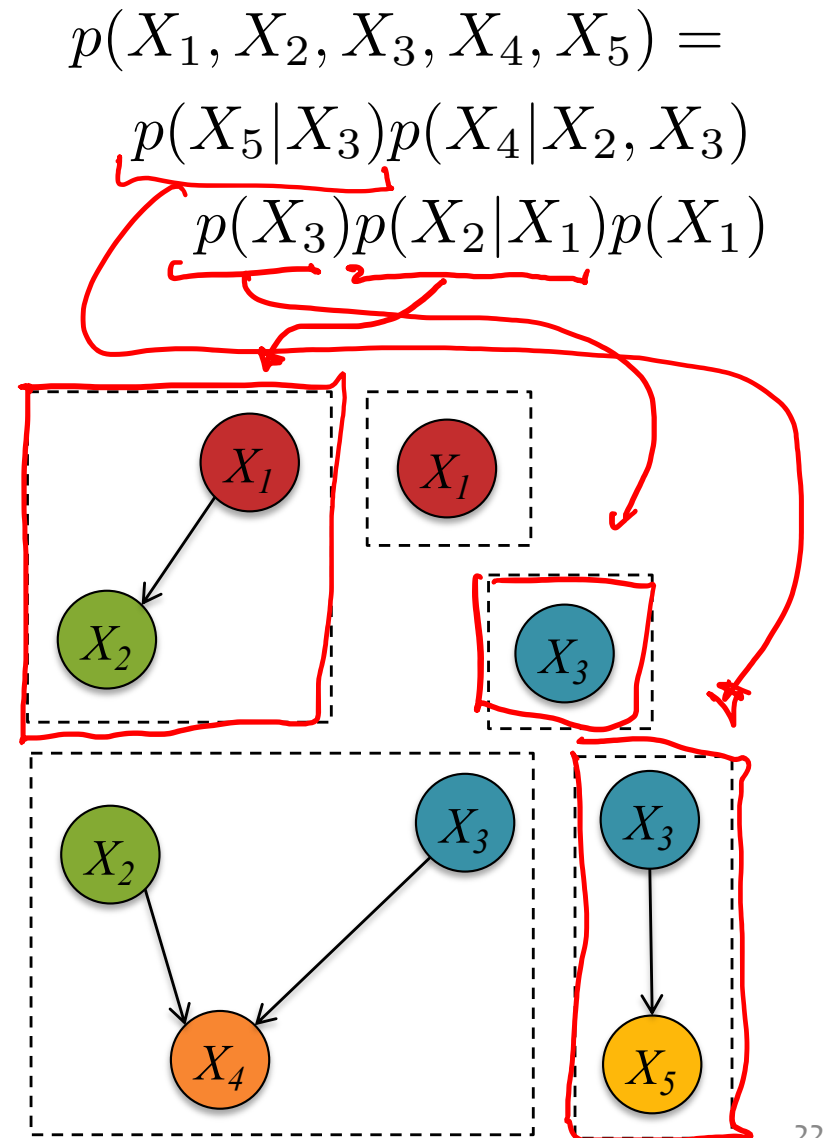
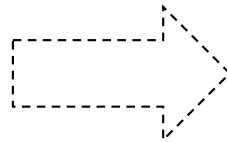
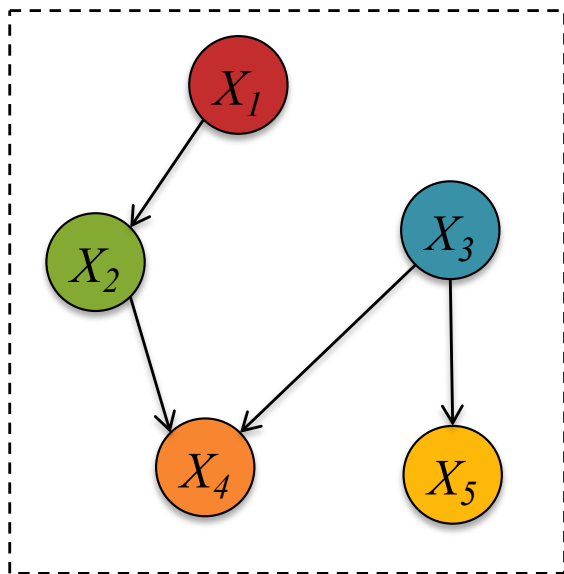


$$p(X_1, X_2, X_3, X_4, X_5) =$$
$$p(X_5|X_3)p(X_4|X_2, X_3)$$
$$p(X_3)p(X_2|X_1)p(X_1)$$

How do we learn these **conditional** and **marginal** distributions for a Bayes Net?

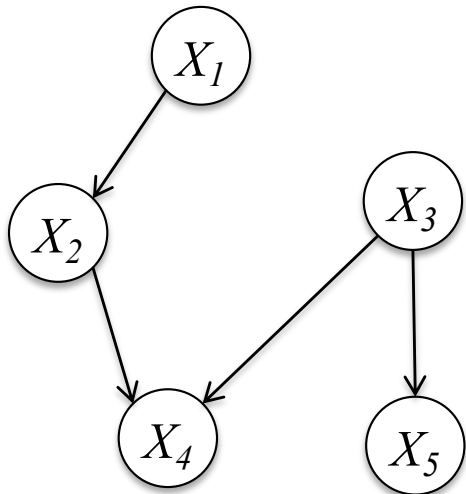
Learning Fully Observed BNs

Learning this fully observed Bayesian Network is **equivalent** to learning five (small / simple) independent networks from the same data



Learning Fully Observed BNs

How do we learn these **conditional** and **marginal** distributions for a Bayes Net?



$$\begin{aligned}\theta^* &= \operatorname{argmax}_{\theta} \log p(X_1, X_2, X_3, X_4, X_5) \\ &= \operatorname{argmax}_{\theta} \log p(X_5|X_3, \theta_5) + \log p(X_4|X_2, X_3, \theta_4) \\ &\quad + \log p(X_3|\theta_3) + \log p(X_2|X_1, \theta_2) \\ &\quad + \log p(X_1|\theta_1)\end{aligned}$$

$$\theta_1^* = \operatorname{argmax}_{\theta_1} \log p(X_1|\theta_1)$$

$$\theta_2^* = \operatorname{argmax}_{\theta_2} \log p(X_2|X_1, \theta_2)$$

$$\theta_3^* = \operatorname{argmax}_{\theta_3} \log p(X_3|\theta_3)$$

$$\theta_4^* = \operatorname{argmax}_{\theta_4} \log p(X_4|X_2, X_3, \theta_4)$$

$$\theta_5^* = \operatorname{argmax}_{\theta_5} \log p(X_5|X_3, \theta_5)$$

Example: Tornado Alarms



1. Imagine that you work at the 911 call center in Dallas
2. You receive six calls informing you that the Emergency Weather Sirens are going off
3. What do you conclude?

Example: Tornado Alarms

Hacking Attack Woke Up Dallas With Emergency Sirens, Officials Say

By ELI ROSENBERG and MAYA SALAM APRIL 8, 2017

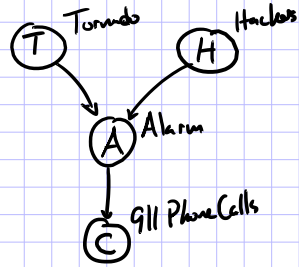


Warning sirens in Dallas, meant to alert the public to emergencies like severe weather, started sounding around 11:40 p.m. Friday, and were not shut off until 1:20 a.m. Rex C. Curry for The New York Times

1. Imagine that you work at the 911 call center in Dallas
2. You receive six calls informing you that the Emergency Weather Sirens are going off
3. What do you conclude?

Learning Fully Observed BNs

Ex: Tornado Alarms $p(\dots) = \prod_j p(x_{ij} | \text{parents}(x_{ij}))$



$H \sim \text{Bernoulli}(\eta)$ ← parameters
 $T \sim \text{Bernoulli}(\tau)$ ← parameters
 $A \sim \text{Bernoulli}(\alpha_{H,T})$ ← no parameters
 $C \sim \text{Uniform}(\{1, \dots, 63\}) + A * \text{Uniform}(\{1, \dots, 63\})$
← integer

Dataset	i	T	H	A	C
	1	0	0	0	2
	2	0	0	0	6
	3	0	0	0	4
	⋮	1	0	0	3
	⋮	1	0	0	1
	⋮	1	0	1	10
	⋮	1	0	1	7
	⋮	0	1	0	2
	⋮	0	1	1	12
	⋮	0	1	0	5
	⋮	1	1	1	10
	12	1	0	0	2

MLEs in Closed Form

$$\begin{aligned}
 \ell(\eta, \tau, \alpha) &= \log \prod_{i=1}^{12} p(t^{(i)}, h^{(i)}, a^{(i)}, c^{(i)} | \eta, \tau, \alpha) \\
 &= \sum_{i=1}^{12} \log p(t^{(i)} | \tau) + \log p(h^{(i)} | \eta) \\
 &\quad + \log p(a^{(i)} | t^{(i)}, h^{(i)}, \alpha) + \log p(c^{(i)} | a^{(i)})
 \end{aligned}$$

$$\hat{\eta}, \hat{\tau}, \hat{\alpha} = \underset{\eta, \tau, \alpha}{\text{argmax}} \ell(\eta, \tau, \alpha)$$

$$\hat{\eta} = \underset{\eta}{\text{argmax}} \sum_{i=1}^{12} \log p(h^{(i)} | \eta) = \#(T=1) / N$$

$$\hat{\tau} = \underset{\tau}{\text{argmax}} \sum_{i=1}^{12} \log p(t^{(i)} | \tau) = \#(H=1) / N$$

$$\hat{\alpha} = \underset{\alpha}{\text{argmax}} \sum_{i=1}^{12} \log p(a^{(i)} | t^{(i)}, h^{(i)}, \alpha)$$

$$\hat{\alpha}_{t,h} = \frac{\#(A=1, T=t, H=h)}{\#(T=t, H=h)}$$

What are the MLEs?

$$\hat{\eta} = 1/3$$

$$\hat{\tau} = 1/2$$

$$\hat{\alpha} = \begin{array}{c|cc} & H=0 & H=1 \\ \hline T=0 & 0 & 1/3 \\ T=1 & 2/3 & 1 \end{array}$$

INFERENCE FOR BAYESIAN NETWORKS

A Few Problems for Bayes Nets

Suppose we already have the parameters of a Bayesian Network...

1. How do we compute the probability of a specific assignment to the variables?

$$P(T=t, H=h, A=a, C=c) = \underline{P(T=t)} \underline{P(H=h)} \underline{P(A=a | H=h, T=t)} \underline{P(C=c | A=a)}$$

2. How do we draw a sample from the joint distribution?

$$t, h, a, c \sim P(T, H, A, C) \quad \text{"ancestral sampling"}$$

3. How do we compute marginal probabilities?

$$P(A) = \dots$$

4. How do we draw samples from a conditional distribution?

$$t, h, a \sim P(T, H, A | C = c)$$

5. How do we compute conditional marginal probabilities?

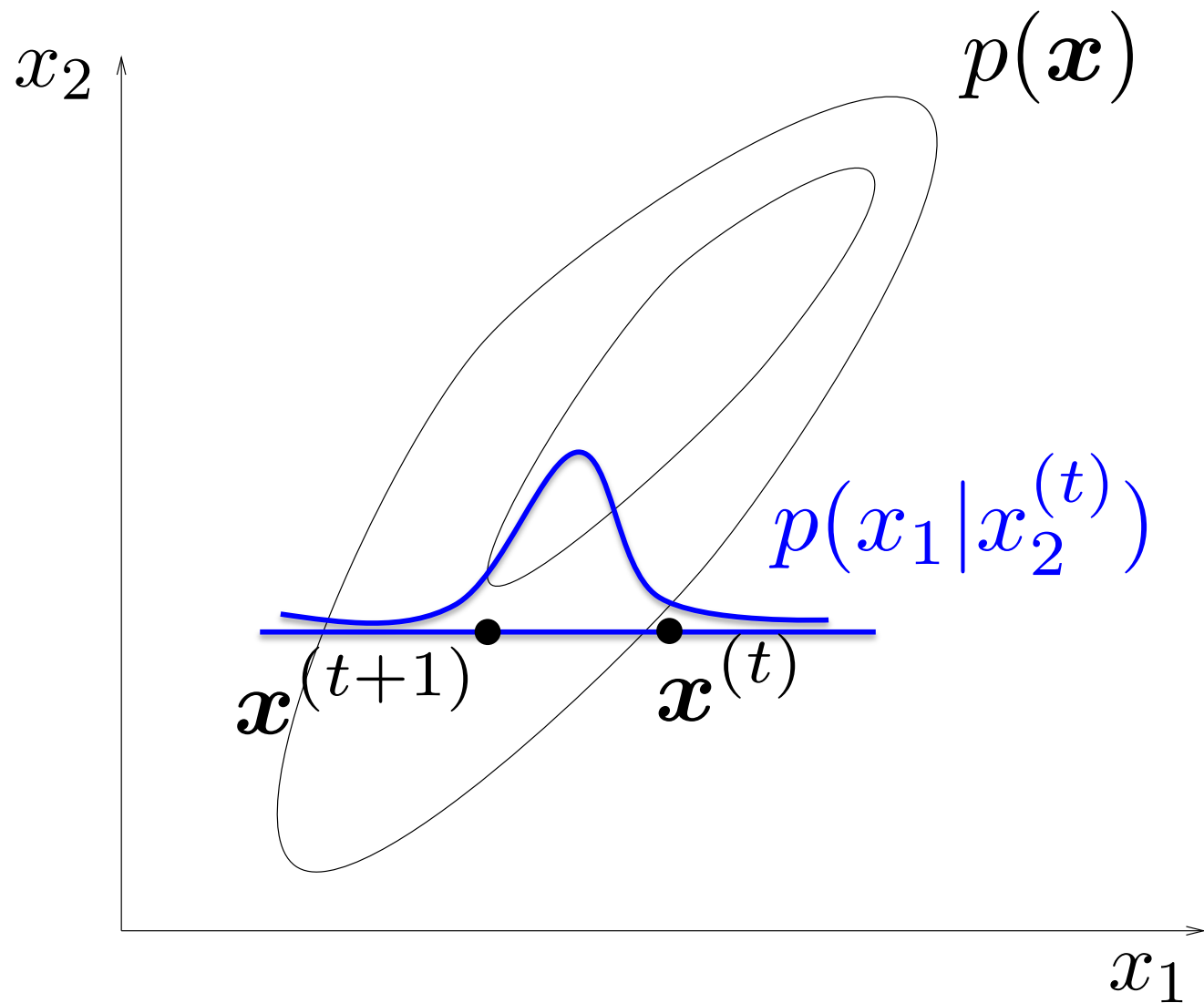
$$P(H | C = c) = \dots$$

T	H	A	C
0	1	1	7
1	0	0	3
1	1	1	10

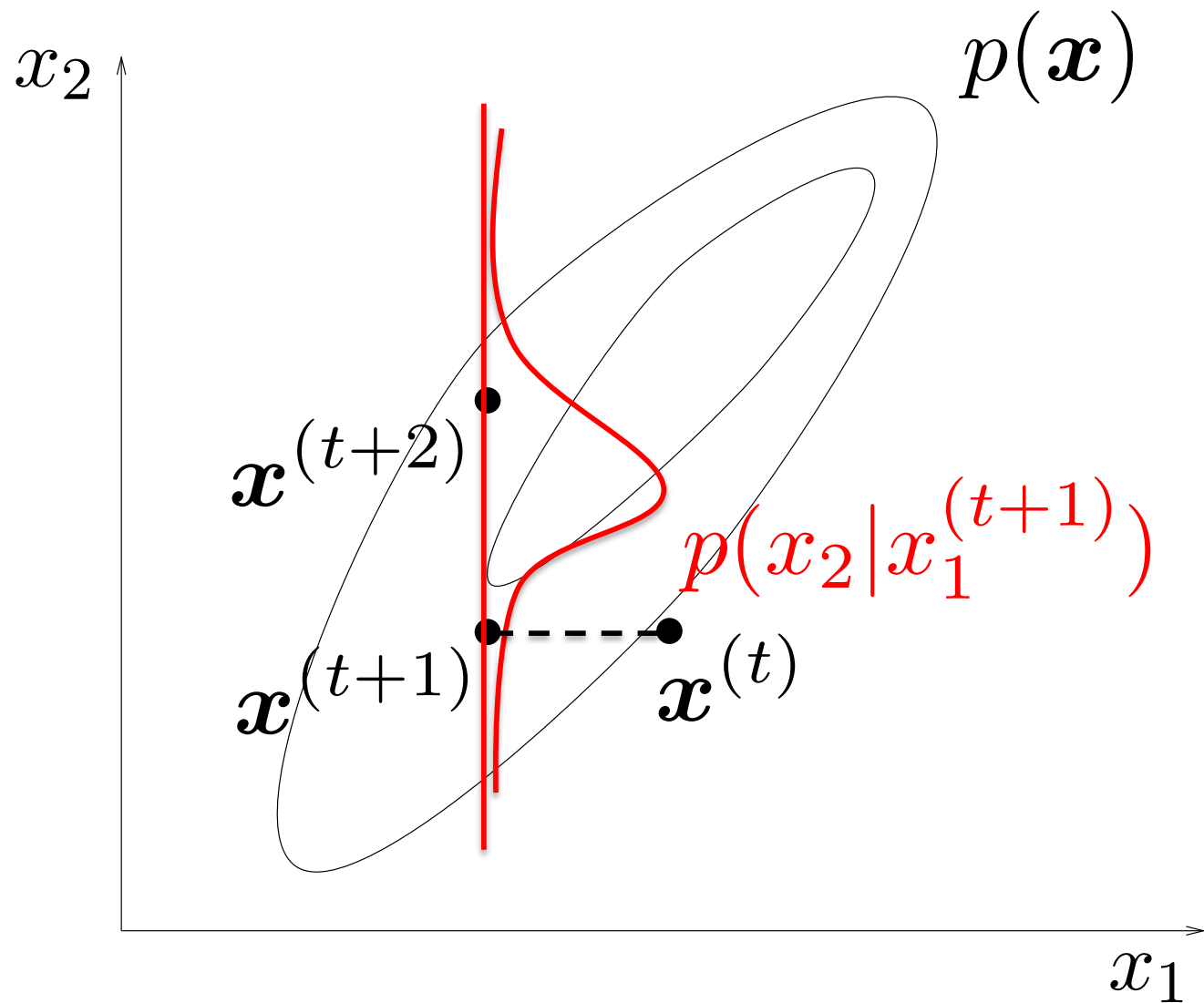
Can we use samples

$P(A=1) = 2/3$
 $P(A=0) = 1/3$
 $P(C=5) = ?$

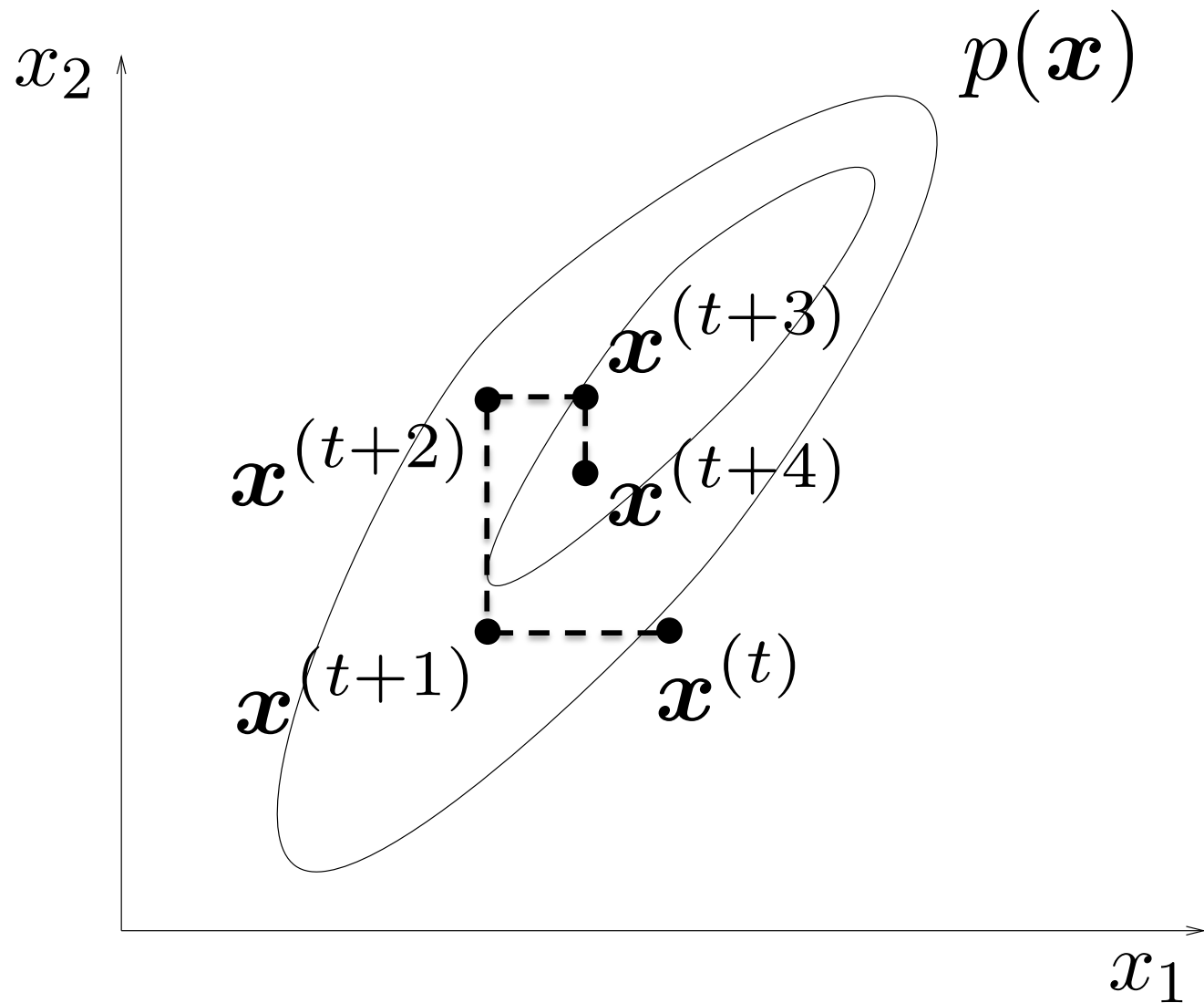
Gibbs Sampling



Gibbs Sampling



Gibbs Sampling



Gibbs Sampling

Question:

How do we draw samples from a conditional distribution?

$$y_1, y_2, \dots, y_J \sim p(y_1, y_2, \dots, y_J \mid x_1, x_2, \dots, x_J)$$

(Approximate) Solution:

- Initialize $y_1^{(0)}, y_2^{(0)}, \dots, y_J^{(0)}$ to arbitrary values
- For $t = 1, 2, \dots$:
 - $y_1^{(t+1)} \sim p(y_1 \mid y_2^{(t)}, \dots, y_J^{(t)}, x_1, x_2, \dots, x_J)$
 - $y_2^{(t+1)} \sim p(y_2 \mid y_1^{(t+1)}, y_3^{(t)}, \dots, y_J^{(t)}, x_1, x_2, \dots, x_J)$
 - $y_3^{(t+1)} \sim p(y_3 \mid y_1^{(t+1)}, y_2^{(t+1)}, y_4^{(t)}, \dots, y_J^{(t)}, x_1, x_2, \dots, x_J)$
 - ...
 - $y_J^{(t+1)} \sim p(y_J \mid y_1^{(t+1)}, y_2^{(t+1)}, \dots, y_{J-1}^{(t+1)}, x_1, x_2, \dots, x_J)$

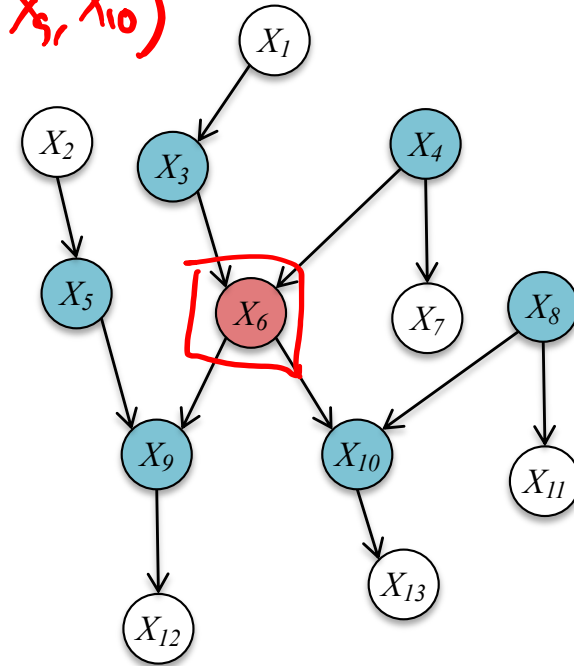
Properties:

- This will eventually yield samples from $p(y_1, y_2, \dots, y_J \mid x_1, x_2, \dots, x_J)$
- But it might take a long time -- just like other Markov Chain Monte Carlo methods

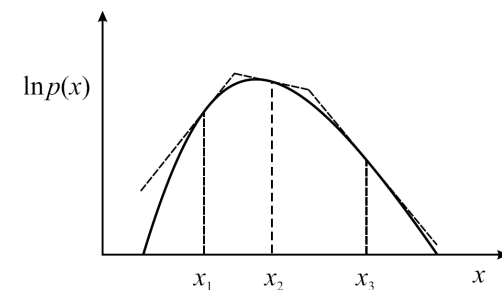
Gibbs Sampling

$$p(x_6 | x_1, x_2, \dots, x_{13}) = p(x_6 | x_3, x_4, x_5, x_8, x_9, x_{10})$$

Full conditionals
only need to
condition on the
Markov
boundary



- Must be “easy” to sample from conditionals
- Many conditionals are log-concave and are amenable to adaptive rejection sampling



Learning Objectives

Bayesian Networks

You should be able to...

1. Identify the conditional independence assumptions given by a generative story or a specification of a joint distribution
2. Draw a Bayesian network given a set of conditional independence assumptions
3. Define the joint distribution specified by a Bayesian network
4. Use domain knowledge to construct a (simple) Bayesian network for a real-world modeling problem
5. Depict familiar models as Bayesian networks
6. Use d-separation to prove the existence of conditional independencies in a Bayesian network
7. Employ a Markov boundary to identify conditional independence assumptions of a graphical model
8. Develop a supervised learning algorithm for a Bayesian network
9. Use samples from a joint distribution to compute marginal probabilities
10. Sample from the joint distribution specified by a generative story
11. Implement a Gibbs sampler for a Bayesian network

eh...

LEARNING PARADIGMS

Learning Paradigms

Paradigm	Data
Supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N$ $\mathbf{x} \sim p^*(\cdot)$ and $y = c^*(\cdot)$
↪ Regression	$y^{(i)} \in \mathbb{R}$
↪ Classification	$y^{(i)} \in \{1, \dots, K\}$
↪ Binary classification	$y^{(i)} \in \{+1, -1\}$
↪ Structured Prediction	$\mathbf{y}^{(i)}$ is a vector

Learning Paradigms

Paradigm	Data
Supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N$ $\mathbf{x} \sim p^*(\cdot)$ and $y = c^*(\cdot)$
↪ Regression	$y^{(i)} \in \mathbb{R}$
↪ Classification	$y^{(i)} \in \{1, \dots, K\}$
↪ Binary classification	$y^{(i)} \in \{+1, -1\}$
↪ Structured Prediction	$\mathbf{y}^{(i)}$ is a vector
Unsupervised	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ $\mathbf{x} \sim p^*(\cdot)$

Learning Paradigms

Paradigm	Data
Supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N \quad \mathbf{x} \sim p^*(\cdot) \text{ and } y = c^*(\cdot)$
↪ Regression	$y^{(i)} \in \mathbb{R}$
↪ Classification	$y^{(i)} \in \{1, \dots, K\}$
↪ Binary classification	$y^{(i)} \in \{+1, -1\}$
↪ Structured Prediction	$\mathbf{y}^{(i)}$ is a vector
Unsupervised	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N \quad \mathbf{x} \sim p^*(\cdot)$
Semi-supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N_1} \cup \{\mathbf{x}^{(j)}\}_{j=1}^{N_2}$

Learning Paradigms

Paradigm	Data
Supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N \quad \mathbf{x} \sim p^*(\cdot) \text{ and } y = c^*(\cdot)$
↪ Regression	$y^{(i)} \in \mathbb{R}$
↪ Classification	$y^{(i)} \in \{1, \dots, K\}$
↪ Binary classification	$y^{(i)} \in \{+1, -1\}$
↪ Structured Prediction	$\mathbf{y}^{(i)}$ is a vector
Unsupervised	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N \quad \mathbf{x} \sim p^*(\cdot)$
Semi-supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N_1} \cup \{\mathbf{x}^{(j)}\}_{j=1}^{N_2}$
Online	$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), (\mathbf{x}^{(3)}, y^{(3)}), \dots\}$

Learning Paradigms

Paradigm	Data
Supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N \quad \mathbf{x} \sim p^*(\cdot) \text{ and } y = c^*(\cdot)$
↪ Regression	$y^{(i)} \in \mathbb{R}$
↪ Classification	$y^{(i)} \in \{1, \dots, K\}$
↪ Binary classification	$y^{(i)} \in \{+1, -1\}$
↪ Structured Prediction	$\mathbf{y}^{(i)}$ is a vector
Unsupervised	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N \quad \mathbf{x} \sim p^*(\cdot)$
Semi-supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N_1} \cup \{\mathbf{x}^{(j)}\}_{j=1}^{N_2}$
Online	$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), (\mathbf{x}^{(3)}, y^{(3)}), \dots\}$
Active Learning	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ and can query $y^{(i)} = c^*(\cdot)$ at a cost

Learning Paradigms

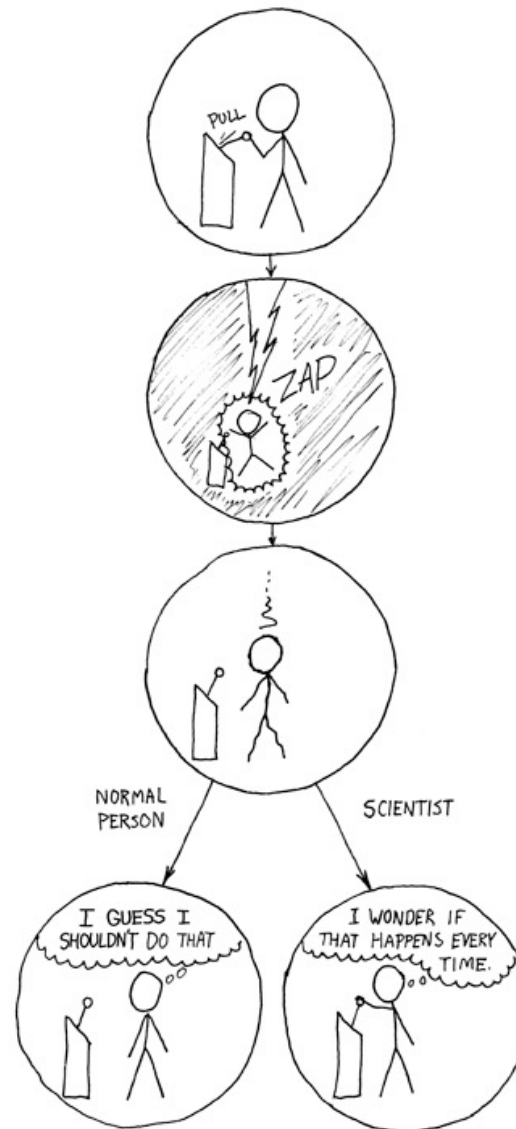
Paradigm	Data
Supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N \quad \mathbf{x} \sim p^*(\cdot) \text{ and } y = c^*(\cdot)$
↪ Regression	$y^{(i)} \in \mathbb{R}$
↪ Classification	$y^{(i)} \in \{1, \dots, K\}$
↪ Binary classification	$y^{(i)} \in \{+1, -1\}$
↪ Structured Prediction	$\mathbf{y}^{(i)}$ is a vector
Unsupervised	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N \quad \mathbf{x} \sim p^*(\cdot)$
Semi-supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N_1} \cup \{\mathbf{x}^{(j)}\}_{j=1}^{N_2}$
Online	$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), (\mathbf{x}^{(3)}, y^{(3)}), \dots\}$
Active Learning	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ and can query $y^{(i)} = c^*(\cdot)$ at a cost
Imitation Learning	$\mathcal{D} = \{(s^{(1)}, a^{(1)}), (s^{(2)}, a^{(2)}), \dots\}$

Learning Paradigms

Paradigm	Data
Supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N \quad \mathbf{x} \sim p^*(\cdot) \text{ and } y = c^*(\cdot)$
↪ Regression	$y^{(i)} \in \mathbb{R}$
↪ Classification	$y^{(i)} \in \{1, \dots, K\}$
↪ Binary classification	$y^{(i)} \in \{+1, -1\}$
↪ Structured Prediction	$\mathbf{y}^{(i)}$ is a vector
Unsupervised	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N \quad \mathbf{x} \sim p^*(\cdot)$
Semi-supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N_1} \cup \{\mathbf{x}^{(j)}\}_{j=1}^{N_2}$
Online	$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), (\mathbf{x}^{(3)}, y^{(3)}), \dots\}$
Active Learning	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ and can query $y^{(i)} = c^*(\cdot)$ at a cost
Imitation Learning	$\mathcal{D} = \{(s^{(1)}, a^{(1)}), (s^{(2)}, a^{(2)}), \dots\}$
Reinforcement Learning	$\mathcal{D} = \{(s^{(1)}, a^{(1)}, r^{(1)}), (s^{(2)}, a^{(2)}, r^{(2)}), \dots\}$

REINFORCEMENT LEARNING

Reinforcement Learning



RL: Examples



Source: <https://www.cnet.com/news/boston-dynamics-robot-dog-spot-finally-goes-on-sale-for-74500/>

Source: <https://techobserver.net/2019/06/argo-ai-self-driving-car-research-center/>

Source: <https://twitter.com/alphagomovie>

Source: <https://www.wired.com/2012/02/high-speed-trading/>



AlphaGo

Source: https://www.youtube.com/watch?v=WXuK6gekU1Y&ab_channel=DeepMind

History of Reinforcement Learning

- Roots in the **psychology of animal learning** (Thorndike, 1911).
- Another independent thread was the problem of **optimal control**, and its solution using **dynamic programming** (Bellman, 1957).
- Idea of **temporal difference** learning (on-line method), e.g., playing board games (Samuel, 1959).
- A major breakthrough was the discovery of **Q-learning** (Watkins, 1989).

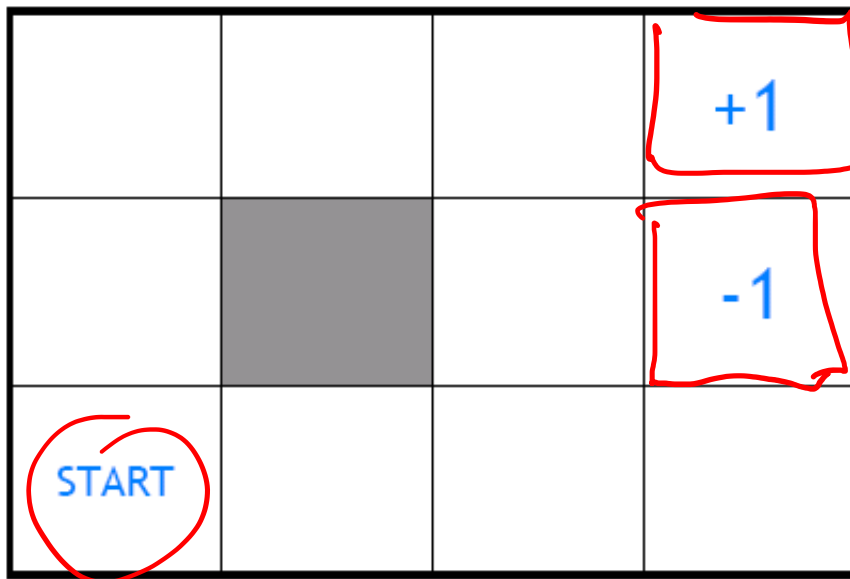
What is special about RL?

- RL is learning how to map states to actions, so as to **maximize** a numerical **reward** over time.
- Unlike other forms of learning, it is a multistage decision-making process (often **Markovian**).
- An RL agent must learn by **trial-and-error**. (Not entirely supervised, but interactive)
- Actions may affect not only the immediate reward but also subsequent rewards (**Delayed effect**).

Elements of RL

- A **policy**
 - A map from **state space** to **action space**.
 - May be stochastic.
- A **reward function**
 - It maps each state (or, state-action pair) to a real number, called **reward**.
- A **value function**
 - Value of a state (or, state-action pair) is the **total expected reward**, starting from that state (or, state-action pair).

Example: Robot in a Room



actions: UP, DOWN, LEFT, RIGHT

UP

80%

10%

10%

move UP

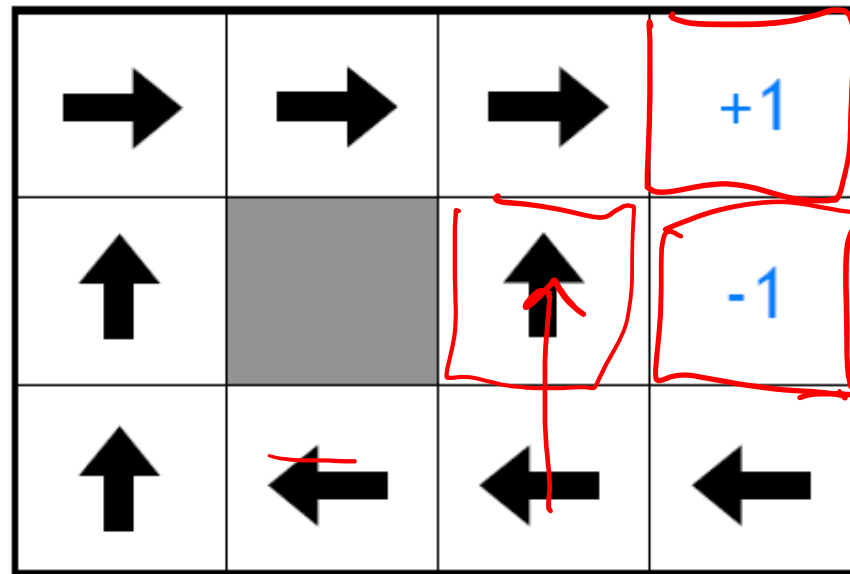
move LEFT

move RIGHT



- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step

Example: Robot in a Room

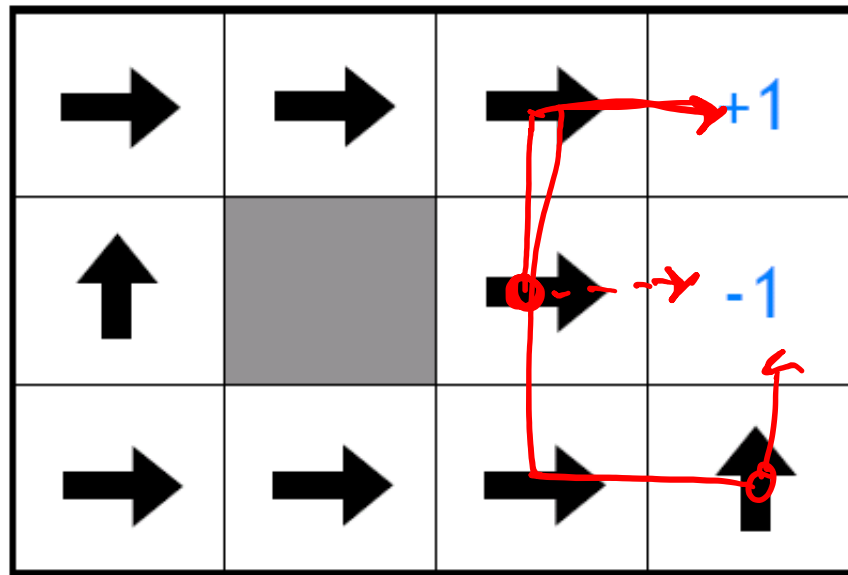


Question: Q2
Is this policy optimal: yes or no? Briefly justify your answer.

Answer: (Hint: both yes and no are acceptable answers, I'm interested in your justification.)

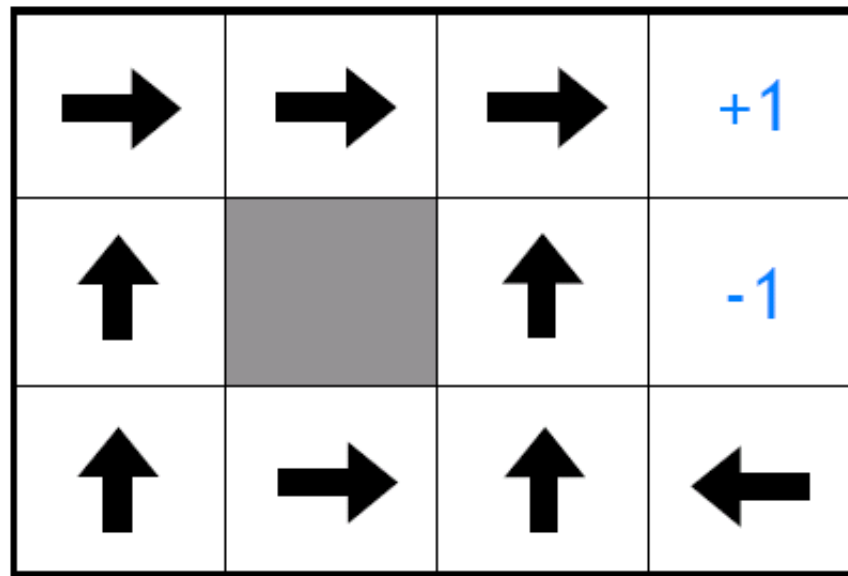
Example: Robot in a Room

- Reward for each step -2



Example: Robot in a Room

- Reward for each step: -0.1



The Precise Goal

- To find a **policy** that maximizes the **Value function**.
 - transitions and rewards usually not available
- There are different approaches to achieve this goal in various situations.

• Value iteration and Policy iteration are two more classic approaches to this problem. But essentially both are **dynamic programming**.

• Q-learning is a more recent approaches to this problem. Essentially it is a **temporal-difference method**.

MARKOV DECISION PROCESSES

RL: Components

From the Environment (i.e. the MDP)

- State space, \mathcal{S}
- Action space, \mathcal{A}
- Reward function, $R(s, a)$, $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$
- Transition probabilities, $p(s' | s, a)$
 - Deterministic transitions:

$$p(s' | s, a) = \begin{cases} 1 & \text{if } \delta(s, a) = s' \\ 0 & \text{otherwise} \end{cases}$$

where $\delta(s, a)$ is a transition function

Markov Assumption

$$p(s_{t+1} | s_t, a_t, \dots, s_1, a_1) \\ = p(s_{t+1} | s_t, a_t)$$

From the Model

- Policy, $\pi : \mathcal{S} \rightarrow \mathcal{A}$
- Value function, $V^\pi : \mathcal{S} \rightarrow \mathbb{R}$
 - Measures the expected total payoff of starting in some state s and executing policy π

Markov Decision Process (MDP)

- For **supervised learning** the **PAC learning framework** provided assumptions about where our data came from:

$$\mathbf{x} \sim p^*(\cdot) \text{ and } y = c^*(\cdot)$$

- For **reinforcement learning** we assume our data comes from a **Markov decision process (MDP)**

Markov Decision Processes (MDP)

In RL, the source of our data is an MDP:

1. Start in some initial state $s_0 \in \mathcal{S}$
2. For time step t :
 1. Agent observes state $s_t \in \mathcal{S}$
 2. Agent takes action $a_t \in \mathcal{A}$ where $a_t = \pi(s_t)$
 3. Agent receives reward $r_t \in \mathbb{R}$ where $r_t = R(s_t, a_t)$
 4. Agent transitions to state $s_{t+1} \in \mathcal{S}$ where $s_{t+1} \sim p(s' | s_t, a_t)$
3. Total reward is $\sum_{t=0}^{\infty} \gamma^t r_t$
 - The value γ is the “discount factor”, a hyperparameter $0 < \gamma < 1$

- Makes the same Markov assumption we used for HMMs! The next state only depends on the current state and action.
- *Def.:* we **execute** a policy π by taking action $a = \pi(s)$ when in state s