10-301/10-601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

HMMs

+

Bayesian Networks

Matt Gormley
Lecture 20
Mar. 29, 2023

Reminders

- Practice Problems: Exam 2
  - Out: Fri, Mar. 24
- Exam 2
  - Thu, Mar. 30, 6:30pm – 8:30pm
- Homework 7: Hidden Markov Models
  - Out: Fri, Mar. 31
  - Due: Mon, Apr. 10 at 11:59pm

THE FORWARD-BACKWARD ALGORITHM

Forward-Backward Algorithm

Definitions

$\alpha_t(k) \triangleq p(x_1, \ldots, x_t, y_t = k)$

$\beta_t(k) \triangleq p(x_{t+1}, \ldots, x_T \mid y_t = k)$

Assume

$y_0 = \text{START}$

$y_{T+1} = \text{END}$

1. Initialize

$$\alpha_0(\text{START}) = 1 \qquad \alpha_0(k) = 0, \; \forall k \neq \text{START}$$

$$\beta_{T+1}(\text{END}) = 1 \qquad \beta_{T+1}(k) = 0, \; \forall k \neq \text{END}$$

2. Forward Algorithm

for $t = 1, \ldots, T+1$:

for $k = 1, \ldots, K$:

$$\alpha_t(k) = \sum_{j=1}^{K} p(x_t \mid y_t = k) \alpha_{t-1}(j) p(y_t = k \mid y_{t-1} = j)$$

3. Backward Algorithm

for $t = T, \ldots, 0$:

for $k = 1, \ldots, K$:

$$\beta_t(k) = \sum_{j=1}^{K} p(x_{t+1} \mid y_{t+1} = j) \beta_{t+1}(j) p(y_{t+1} = j \mid y_t = k)$$

4. Evaluation $p(\mathbf{x}) = \alpha_{T+1}(\text{END})$

5. Marginals $p(y_t = k \mid \mathbf{x}) = \frac{\alpha_t(k)\beta_t(k)}{p(\mathbf{x})}$

Forward-Backward Algorithm

Definitions

$\alpha_t(k) \triangleq p(x_1, \ldots, x_t, y_t = k)$

$\beta_t(k) \triangleq p(x_{t+1}, \ldots, x_T \mid y_t = k)$

Assume

$y_0 = \text{START}$

$y_{T+1} = \text{END}$

1. Initialize

$\alpha_0(\text{START}) = 1 \qquad \alpha_0(k) = 0, \ \forall k \neq \text{START}$

$\beta_{T+1}(\text{END}) = 1 \qquad \beta_{T+1}(k) = 0, \ \forall k \neq \text{END}$

2. Forward Algorithm

for $t = 1, \ldots, T+1$:

for $k = 1, \ldots, K$:

$$\alpha_t(k) = \sum_{j=1}^{K} p(x_t \mid y_t = k)\alpha_{t-1}(j)p(y_t = k \mid y_{t-1} = j)$$

O(K²T)

O(K)

kward Algorithm

for $t = T, \ldots, 0$:

or $k = 1, \ldots, K$:

$$\beta_t(k) = \sum_{j=1}^{K} p(x_{t+1} \mid y_{t+1} = j)\beta_{t+1}(j)p(y_{t+1} = j \mid y_t = k)$$

Brute force algorithm would be $O(K^T)$

4. Evaluation $p(\mathbf{x}) = \alpha_{T+1}(\text{END})$

5. Marginals $p(y_t = k \mid \mathbf{x}) = \frac{\alpha_t(k)\beta_t(k)}{p(\mathbf{x})}$

Derivation of Forward Algorithm

Definition:

$$\alpha_t(k) \triangleq p(x_1, \ldots, x_t, y_t = k)$$

Derivation:

$$\begin{aligned}
\alpha_T(\text{END}) &= p(x_1, \ldots, x_T, y_T = \text{END}) && y_T \text{ as shorthand for } y_T = \text{END} \\
&= p(x_1, \ldots, x_T \mid y_T) p(y_T) && \text{by chain rule} \\
&= p(x_T \mid y_T) p(x_1, \ldots, x_{T-1} \mid y_T) p(y_T) && \text{by cond indep of HMM} \\
&= p(x_T \mid y_T) p(x_1, \ldots, x_{T-1}, y_T) && \text{by rev chain rule} \\
&= p(x_T \mid y_T) \sum_{y_{T-1}} p(x_1, \ldots, x_{T-1}, y_{T-1}, y_T) && \text{by def of marginal} \\
&= p(x_T \mid y_T) \sum_{y_{T-1}} p(x_1, \ldots, x_{T-1}, y_T \mid y_{T-1}) p(y_{T-1}) && \text{by chain rule} \\
&= p(x_T \mid y_T) \sum_{y_{T-1}} p(x_1, \ldots, x_{T-1} \mid y_T) p(y_T \mid y_{T-1}) p(y_{T-1}) && \text{by cond indept of HMM} \\
&= p(x_T \mid y_T) \sum_{y_{T-1}} p(x_1, \ldots, x_{T-1}, y_{T-1}) p(y_T \mid y_{T-1}) && \text{by rev chain rule} \\
&= p(x_T \mid y_T) \sum_{y_{T-1}} \alpha_{T-1}(y_{T-1}) p(y_T \mid y_{T-1}) && \text{by def of } \alpha
\end{aligned}$$

FORWARD-BACKWARD IN LOG SPACE

Forward-Backward Algorithm

De

α_t

β_t

As

y_0

y_T

Problem:
Implementing F-B as shown here could run into **underflow** (i.e. floating point precision issues).

Why?
Because the algorithm is still multiplying O(T) probabilities together. Each probability is in [0,1] and so their product can get very small.

One solution:
work in log-space!

1. Initialize

$$\alpha_0(\text{START}) = 1 \qquad \alpha_0(k) = 0,\ \forall k \neq \text{START}$$

$$\beta_{T+1}(\text{END}) = 1 \qquad \beta_{T+1}(k) = 0,\ \forall k \neq \text{END}$$

Forward Algorithm

for $t = 1, \ldots, T+1$:

for $k = 1, \ldots, K$:

$$\alpha_t(k) = \sum_{j=1}^{K} \boxed{p(x_t \mid y_t = k)\alpha_{t-1}(j)p(y_t = k \mid y_{t-1} = j)}$$

Backward Algorithm

for $t = T, \ldots, 0$:

for $k = 1, \ldots, K$:

$$\beta_t(k) = \sum_{j=1}^{K} \boxed{p(x_{t+1} \mid y_{t+1} = j)\beta_{t+1}(j)p(y_{t+1} = j \mid y_t = k)}$$

4. Evaluation $p(\mathbf{x}) = \alpha_{T+1}(\text{END})$

5. Marginals $p(y_t = k \mid \mathbf{x}) = \frac{\alpha_t(k)\beta_t(k)}{p(\mathbf{x})}$

Log-space Arithmetic

Log-space Multiplication

- Suppose you wish to multiply two probabilities p_a and p_b together to get $p_c = p_a\, p_b$
- Yet, you want to represent all those numbers as the log of their value:
  - $o_a = \log(p_a)$
  - $o_b = \log(p_b)$
  - $o_c = \log(p_c)$
- To compute o_c from o_a and o_b we simply add them:

  $$\begin{aligned} o_c &= o_a + o_b \\ &= \log(p_a) + \log(p_b) \\ &= \log(p_a\, p_b) \\ &= \log(p_c) \end{aligned}$$

Log-space Addition

- Suppose you wish to add two probabilities p_a and p_b together to get $p_d = p_a + p_b$, yet all in log-space (e.g. $o_d = \log(p_d)$)
- To compute compute o_d from o_a and o_b we must be more careful:

  $$\begin{aligned} o_d &= \text{log-sum-exp}(o_a, o_b) \\ &= \log(\exp(o_a) + \exp(o_b)) \end{aligned}$$

- **Problem:** if we merely implement log-sum-exp as above, we'll probably run into underflow again b/c:
  - $p_a = \exp(o_a)$
  - $p_b = \exp(o_b)$

Log-space Arithmetic

A careful implementation:

```
def log-sum-exp(x_1, ..., x_N):
    c = max(x_1, ..., x_N)
    y = c + log sum_{n=1}^{N} exp(x_n - c)
    return y
```

$$\textbf{def}\ \text{log-sum-exp}(x_1, \ldots, x_N):$$
$$c = \max(x_1, \ldots, x_N)$$
$$y = c + \log \textstyle\sum_{n=1}^{N} \exp(x_n - c)$$
$$\textbf{return}\ y$$

Why does this work?

$$y = \log \sum_{n=1}^{N} \exp(x_n)$$

$$\Rightarrow \exp(y) = \sum_{n=1}^{N} \exp(x_n)$$

$$\Rightarrow \exp(y) = \frac{\exp(c)}{\exp(c)} \sum_{n=1}^{N} \exp(x_n)$$

$$\Rightarrow \exp(y) = \exp(c) \sum_{n=1}^{N} \exp(x_n - c)$$

$$\Rightarrow y = c + \log \sum_{n=1}^{N} \exp(x_n - c)$$

Log-space Addition

- Suppose you wish to add two probabilities p_a and p_b together to get $p_d = p_a + p_b$, yet all in log-space (e.g. $o_d = \log(p_d)$)
- To compute compute o_d from o_a and o_b we must be more careful:

  o_d = log-sum-exp(o_a, o_b)
  $= \log(\exp(o_a) + \exp(o_b))$

- **Problem:** if we merely implement log-sum-exp as above, we'll probably run into underflow again b/c:
  - $p_a = \exp(o_a)$
  - $p_b = \exp(o_b)$

Forward Algorithm (in log-space)

We can run the forward algorithm in log-space using log-multiplication and log-addition. The backward algorithm is analogous.

Definitions

$\log \alpha_t(k) \triangleq \log p(x_1, \ldots, x_t, y_t = k)$

Assume

$y_0 =$ START

1. Initialize

$$\log \alpha_0(\text{START}) = 0 \qquad \log \alpha_0(k) = -\infty, \forall k \neq \text{START}$$

2. Forward Algorithm

for $t = 1, \ldots, T+1$:

for $k = 1, \ldots, K$:

for $j = 1, \ldots, K$:

$o_j = \log p(x_t \mid y_t = k) + \log \alpha_{t-1}(j) + \log p(y_t = k \mid y_{t-1} = j)$

$\log \alpha_t(k) = \text{log-sum-exp}(o_1, \ldots, o_K)$

3. Evaluation $\log p(\mathbf{x}) = \log \alpha_{T+1}(\text{END})$

THE VITERBI ALGORITHM

Inference for HMMs

Whiteboard

- Viterbi algorithm
  (edge weights version)

Viterbi Algorithm

Definitions

$$\omega_t(k) \triangleq \max_{y_1,\ldots,y_{t-1}} p(x_1,\ldots,x_t,y_1,\ldots,y_{t-1},y_t = k)$$

$$b_t(k) \triangleq \operatorname*{argmax}_{y_1,\ldots,y_{t-1}} p(x_1,\ldots,x_t,y_1,\ldots,y_{t-1},y_t = k)$$

Assume

$y_0 = \text{START}$

$y_{T+1} = \text{END}$

1. Initialize

$$\omega_0(\text{START}) = 1 \qquad\qquad \omega_0(k) = 0,\ \forall k \neq \text{START}$$

2. Viterbi Algorithm

for $t = 1,\ldots,T+1$:

for $k = 1,\ldots,K$:

$$\omega_t(k) = \max_{j\in\{1,\ldots,K\}} p(x_t \mid y_t = k)\omega_{t-1}(j)p(y_t = k \mid y_{t-1} = j)$$

$$b_t(k) = \operatorname*{argmax}_{j\in\{1,\ldots,K\}} p(x_t \mid y_t = k)\omega_{t-1}(j)p(y_t = k \mid y_{t-1} = j)$$

3. Compute Most Probable Assignment

$$\hat{y}_T = b_{T+1}(\text{END})$$

for $t = T,\ldots,1$:

$$\hat{y}_t = b_{t+1}(\hat{y}_{t+1})$$

Viterbi Algorithm

Definitions

$$\omega_t(k) \triangleq \max_{y_1,\ldots,y_{t-1}} p(x_1,\ldots,x_t,y_1,\ldots,y_{t-1},y_t=k)$$

$$b_t(k) \triangleq \operatorname*{argmax}_{y_1,\ldots,y_{t-1}} p(x_1,\ldots,x_t,y_1,\ldots,y_{t-1},y_t=k)$$

Assume

$y_0 = \text{START}$

$y_{T+1} = \text{END}$

1. Initialize

$$\omega_0(\text{START}) = 1 \qquad\qquad \omega_0(k) = 0,\ \forall k \neq \text{START}$$

2. Viterbi Algorithm

for $t = 1,\ldots,T+1$:

for $k = 1,\ldots,K$:

$$\omega_t(k) = \max_{j\in\{1,\ldots,K\}} p(x_t \mid y_t = k)\omega_{t-1}(j)p(y_t = k \mid y_{t-1} = j)$$

$$b_t(k) = \operatorname*{argmax}_{j\in\{1,\ldots,K\}} p(x_t \mid y_t = k)\omega_{t-1}(j)p(y_t = k \mid y_{t-1} = j)$$

$O(K^2T)$

3. Compute Most Probable Assignment

$\hat{y}_T = b_{T+1}(\text{END})$

for $t = T,\ldots,1$:

$\hat{y}_t = b_{t+1}(\hat{y}_{t+1})$

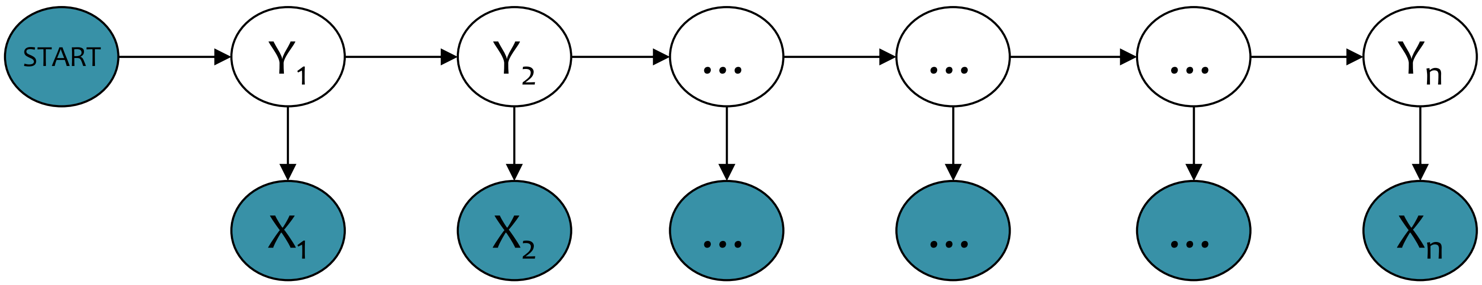
Brute force algorithm would be $O(K^T)$

Inference in HMMs

What is the **computational complexity** of inference for HMMs?

- The **naïve** (brute force) computations for *Evaluation*, *Decoding*, and *Marginals* take **exponential time**, $O(K^T)$
- The **forward-backward** algorithm and **Viterbi** algorithm run in **polynomial time**, $O(T*K^2)$
  - Thanks to dynamic programming!

Shortcomings of Hidden Markov Models



- HMM models capture dependences between each state and only its corresponding observation
  - NLP example: In a sentence segmentation task, each segmental state may depend not just on a single word (and the adjacent segmental stages), but also on the (non-local) features of the whole line such as line length, indentation, amount of white space, etc.
- Mismatch between learning objective function and prediction objective function
  - HMM learns a joint distribution of states and observations P(**Y**, **X**), but in a prediction task, we need the conditional probability P(**Y**|**X**)

MBR DECODING

Inference for HMMs

Four

– ~~Three~~ Inference Problems for an HMM

1. Evaluation: Compute the probability of a given sequence of observations
2. Viterbi Decoding: Find the most-likely sequence of hidden states, given a sequence of observations
3. Marginals: Compute the marginal distribution for a hidden state, given a sequence of observations
4. MBR Decoding: Find the lowest loss sequence of hidden states, given a sequence of observations (Viterbi decoding is a special case)

Minimum Bayes Risk Decoding

- Suppose we given a loss function $l(\boldsymbol{y}', \boldsymbol{y})$ and are asked for a single tagging
- How should we choose just one from our probability distribution $p(\boldsymbol{y}|\boldsymbol{x})$?
- A minimum Bayes risk (MBR) decoder $h(\boldsymbol{x})$ returns the variable assignment with minimum **expected** loss under the model's distribution

$$
\begin{aligned}
h_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \underset{\hat{\boldsymbol{y}}}{\operatorname{argmin}} \; \mathbb{E}_{\boldsymbol{y} \sim p_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}[\ell(\hat{\boldsymbol{y}}, \boldsymbol{y})] \\
&= \underset{\hat{\boldsymbol{y}}}{\operatorname{argmin}} \; \sum_{\boldsymbol{y}} p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) \ell(\hat{\boldsymbol{y}}, \boldsymbol{y})
\end{aligned}
$$

Minimum Bayes Risk Decoding

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \operatorname*{argmin}_{\hat{\boldsymbol{y}}} \; \mathbb{E}_{\boldsymbol{y} \sim p_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{x})}[\ell(\hat{\boldsymbol{y}}, \boldsymbol{y})]$$

Consider some example loss functions:

The ***0-1* loss function** returns *0* only if the two assignments are identical and *1* otherwise:

$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}) = 1 - \mathbb{I}(\hat{\boldsymbol{y}}, \boldsymbol{y})$$

The MBR decoder is:

$$\begin{aligned} h_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \operatorname*{argmin}_{\hat{\boldsymbol{y}}} \; \sum_{\boldsymbol{y}} p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x})(1 - \mathbb{I}(\hat{\boldsymbol{y}}, \boldsymbol{y})) \\ &= \operatorname*{argmax}_{\hat{\boldsymbol{y}}} \; p_{\boldsymbol{\theta}}(\hat{\boldsymbol{y}} \mid \boldsymbol{x}) \end{aligned}$$

which is exactly the Viterbi decoding problem!

Minimum Bayes Risk Decoding

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \underset{\hat{\boldsymbol{y}}}{\text{argmin}} \ \mathbb{E}_{\boldsymbol{y} \sim p_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{x})}[\ell(\hat{\boldsymbol{y}}, \boldsymbol{y})]$$

Consider some example loss functions:

The **Hamming loss** corresponds to accuracy and returns the number of incorrect variable assignments:

$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \sum_{i=1}^{V} (1 - \mathbb{I}(\hat{y}_i, y_i))$$

The MBR decoder is:

$$\hat{y}_i = h_{\boldsymbol{\theta}}(\boldsymbol{x})_i = \underset{\hat{y}_i}{\text{argmax}} \ p_{\boldsymbol{\theta}}(\hat{y}_i \mid \boldsymbol{x})$$

This decomposes across variables and requires the variable marginals.

TO HMMS AND BEYOND…

Unsupervised Learning for HMMs

- Unlike **discriminative** models p(y|x), **generative** models p(x,y) can maximize the likelihood of the data D = {$x^{(1)}$, $x^{(2)}$, … , $x^{(N)}$} where we don't observe any y's.
- This **unsupervised learning** setting can be achieved by finding parameters that maximize the **marginal likelihood**
- We optimize using the **Expectation-Maximization** algorithm

Since we don't observe $\mathbf{y}$, we define the marginal probability:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{Y}} p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) \quad (1)$$

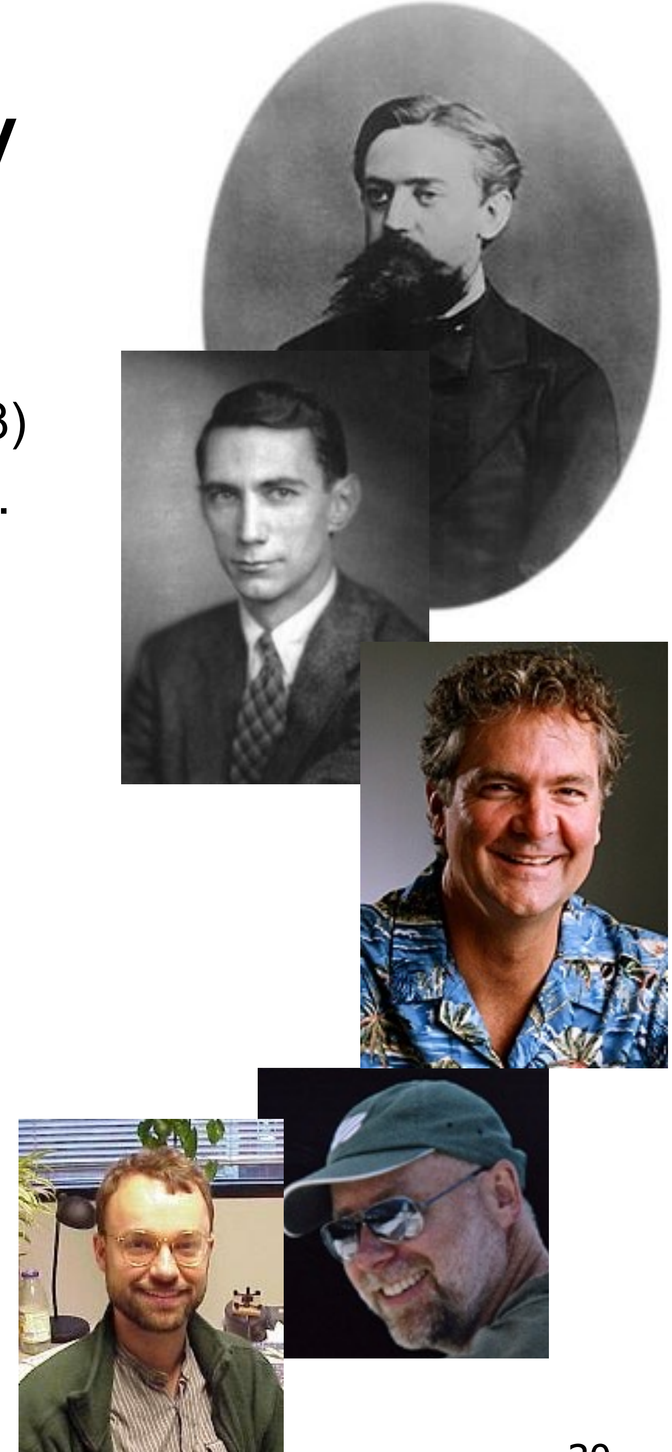
Beyond the scope of today's lecture!

The log-likelihood of the data is thus:

$$\begin{aligned} \ell(\boldsymbol{\theta}) &= \log \prod_{i=1}^{N} p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \\ &= \sum_{i=1}^{N} \log \sum_{\mathbf{y} \in \mathcal{Y}} p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, \mathbf{y}) \quad (3) \end{aligned}$$

HMMs: History

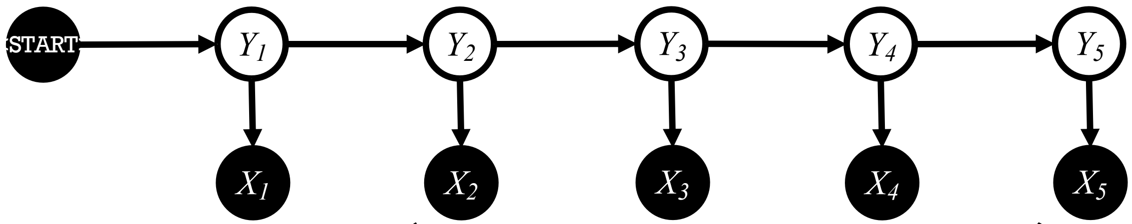
- Markov chains: Andrey Markov (1906)
  - Random walks and Brownian motion
- Used in Shannon's work on information theory (1948)
- Baum-Welsh learning algorithm: late 60's, early 70's.
  - Used mainly for speech in 60s-70s.
- Late 80's and 90's: David Haussler (major player in learning theory in 80's) began to use HMMs for modeling biological sequences
- Mid-late 1990's: Dayne Freitag/Andrew McCallum
  - Freitag thesis with Tom Mitchell on IE from Web using logic programs, grammar induction, etc.
  - McCallum: multinomial Naïve Bayes for text
  - With McCallum, IE using HMMs on CORA
- …



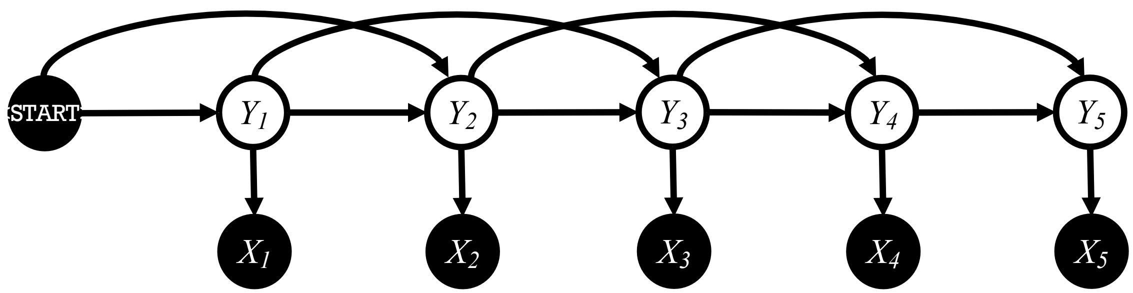
Slide from William Cohen

Higher-order HMMs

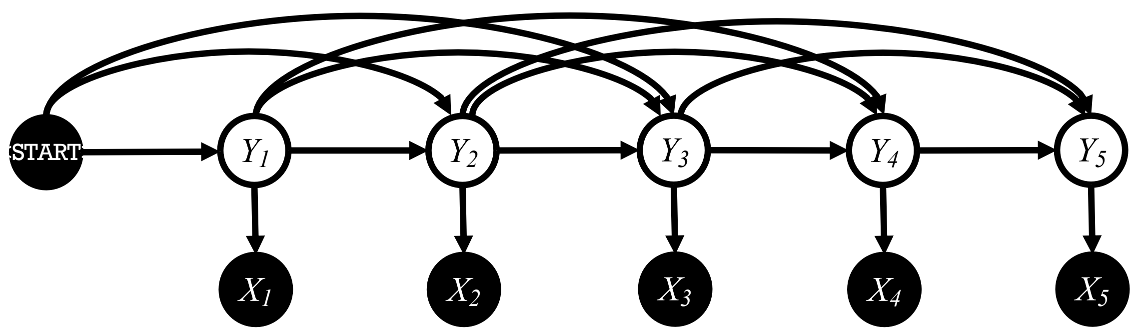
- 1st-order HMM (i.e. bigram HMM)



- 2nd-order HMM (i.e. trigram HMM)

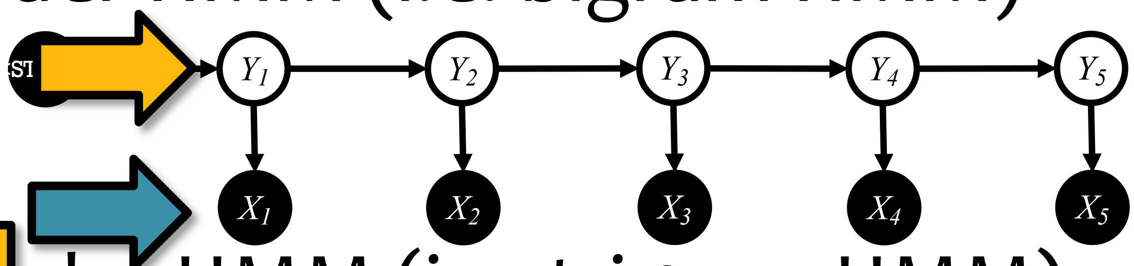


- 3rd-order HMM

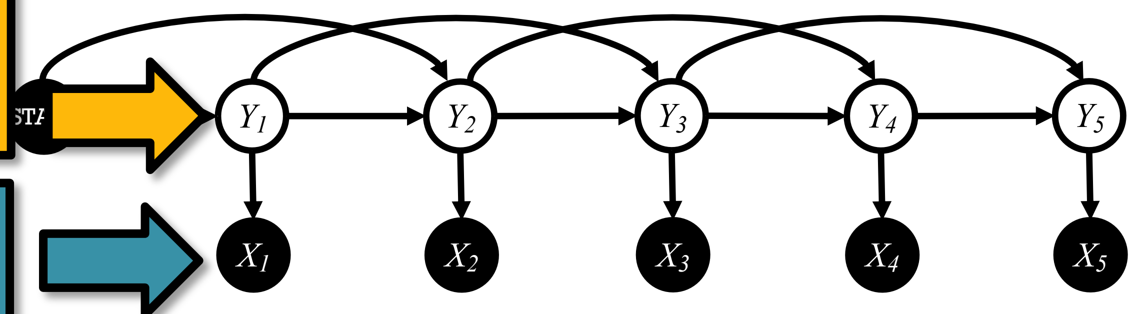


Higher-order HMMs

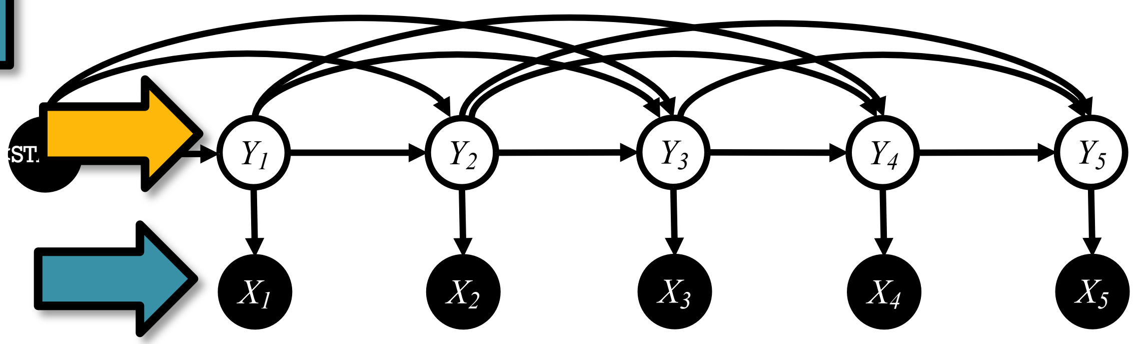
- 1^{st}-order HMM (i.e. bigram HMM)



- rder HMM (i.e. trigram HMM)



- rder HMM



Hidden States, **y**

Observa -tions, **x**

Learning Objectives

Hidden Markov Models

You should be able to…

1. Show that structured prediction problems yield high-computation inference problems
2. Define the first order Markov assumption
3. Draw a Finite State Machine depicting a first order Markov assumption
4. Derive the MLE parameters of an HMM
5. Define the three key problems for an HMM: evaluation, decoding, and marginal computation
6. Derive a dynamic programming algorithm for computing the marginal probabilities of an HMM
7. Interpret the forward-backward algorithm as a message passing algorithm
8. Implement supervised learning for an HMM
9. Implement the forward-backward algorithm for an HMM
10. Implement the Viterbi algorithm for an HMM
11. Implement a minimum Bayes risk decoder with Hamming loss for an HMM

Bayesian Networks

DIRECTED GRAPHICAL MODELS

Example: CMU Mission Control

Bloomberg

Subscribe

Businessweek | Technology

College Students Are About to Put a Robot on the Moon Before NASA

A commercial spaceflight in May will take a Carnegie Mellon-designed rover, named Iris, to the lunar surface.

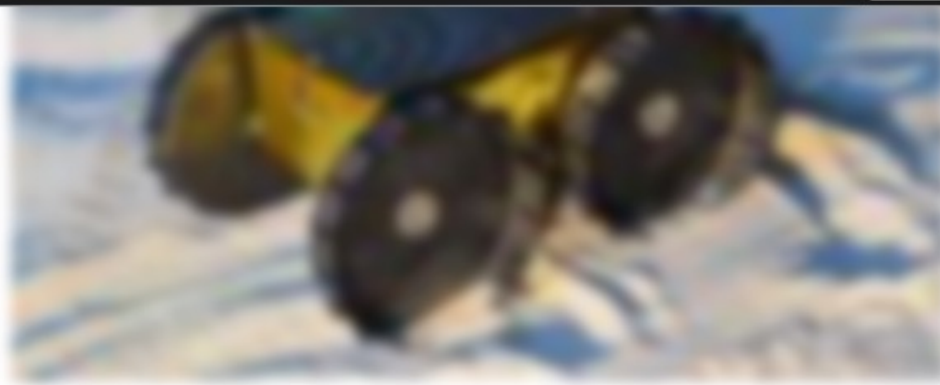
An engineering model of the Iris rover at Carnegie Mellon University's Robotics Institute. *Source: Carnegie Mellon University*

By Katrina Manson
March 29, 2023 at 8:00 AM EDT

Exa trol

Bloomberg

Subscribe



An engineering model of the Iris rover at Carnegie Mellon University's Robotics Institute. *Source: Carnegie Mellon University*

By Katrina Manson
March 29, 2023 at 8:00 AM EDT

Share this article

Gift this article

Follow the authors

@KatrinaManson

+ Get alerts for Katrina Manson

The US was the first and only country to put humans on the moon, but NASA has been noticeably absent from the international competition to put robots there. The USSR landed the first lunar robotic rovers in the 1970s; India tried and failed to land one in 2019. The only lunar rover in operation is China's Yutu-2, a 300-pound machine that's spent the past four years prowling almost two-thirds of a mile across the moon's far side, sending back images of rocks. Greece, Japan and the United Arab Emirates are among those working on their own lunar rover programs.

It seems likely that NASA's robots will also be beaten by a group made up primarily of students at Carnegie Mellon University. About 300 students worked on a rover named Iris

Already a subscriber or Bloomberg Anywhere client? Sign In

Subscribe now for unlimited access to Bloomberg.com and the Bloomberg app

Global news that uncovers a new tomorrow. Cancel anytime.

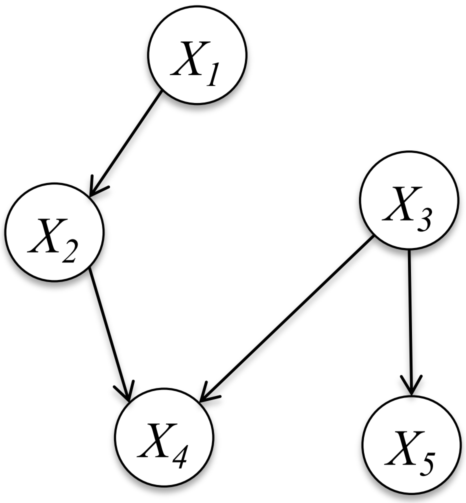
Claim This Offer

Directed Graphical Models (Bayes Nets)

Whiteboard

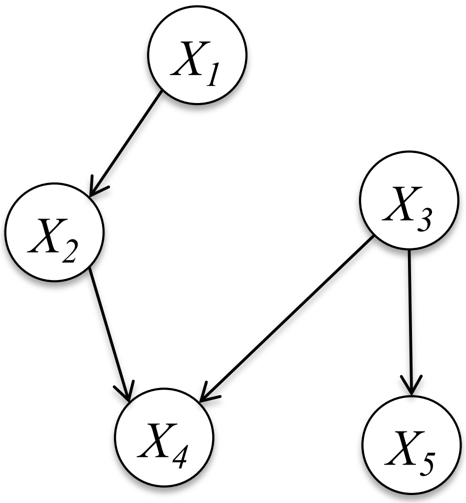
- Example: CMU Mission Control
- Writing Joint Distributions
  - Idea #1: Giant Table
  - Idea #2: Rewrite using chain rule
  - Idea #3: Assume full independence
  - Idea #4: Drop variables from RHS of conditionals
- Definition: Bayesian Network

Bayesian Network



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

Bayesian Network



Definition:

$$P(X_1, \ldots, X_T) = \prod_{t=1}^{T} P(X_t \mid \text{parents}(X_t))$$

- A Bayesian Network is a **directed graphical model**
- It consists of a graph **G** and the conditional probabilities **P**
- These two parts full specify the distribution:
  - Qualitative Specification: **G**
  - Quantitative Specification: **P**

Qualitative Specification

- Where does the qualitative specification come from?
  - Prior knowledge of causal relationships
  - Prior knowledge of modular relationships
  - Assessment from experts
  - Learning from data (i.e. structure learning)
  - We simply prefer a certain architecture (e.g. a layered graph)
  - …

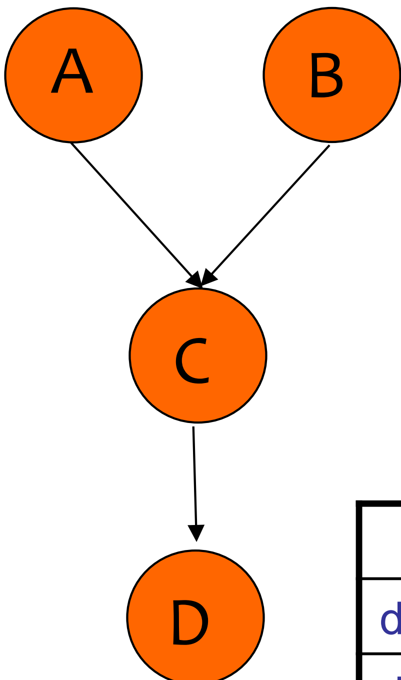
Quantitative Specification

Example: Conditional probability tables (CPTs)
for discrete random variables

$$P(a,b,c.d) = P(a)P(b)P(c|a,b)P(d|c)$$

a^0	0.75
a^1	0.25

b^0	0.33
b^1	0.67



	a^0b^0	a^0b^1	a^1b^0	a^1b^1
c^0	0.45	1	0.9	0.7
c^1	0.55	0	0.1	0.3

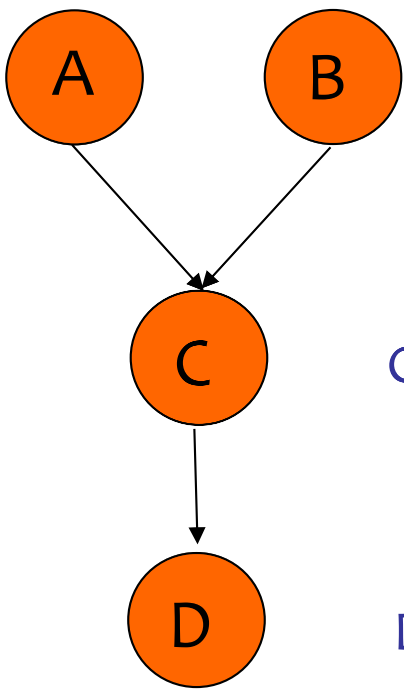
	c^0	c^1
d^0	0.3	0.5
d^1	07	0.5

Quantitative Specification

Example: Conditional probability density functions (CPDs) for continuous random variables

$A \sim N(\mu_a, \Sigma_a)$ $B \sim N(\mu_b, \Sigma_b)$

$$P(a,b,c.d) = P(a)P(b)P(c|a,b)P(d|c)$$



$C \sim N(A+B, \Sigma_c)$

$D \sim N(\mu_d + C, \Sigma_d)$

P(D| C)

D

C

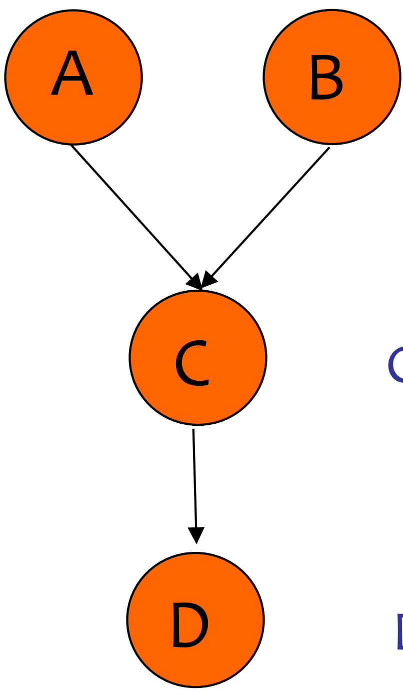
Quantitative Specification

Example: Combination of CPTs and CPDs for a mix of discrete and continuous variables

a^0	0.75
a^1	0.25

b^0	0.33
b^1	0.67

$$P(a,b,c.d) = P(a)P(b)P(c|a,b)P(d|c)$$



$C \sim N(A+B, \Sigma_c)$

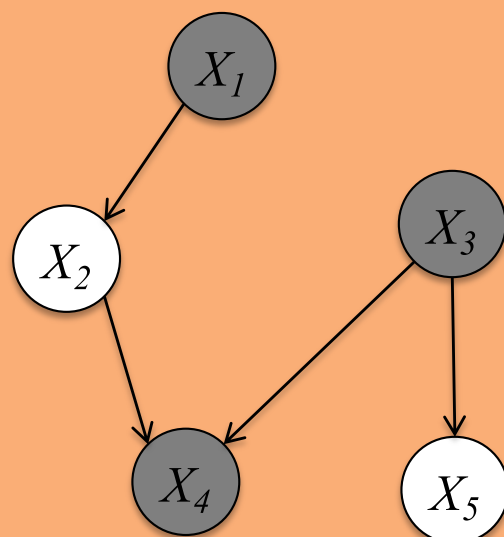
$D \sim N(\mu_d + C, \Sigma_d)$

Observed Variables

- In a graphical model, **shaded nodes** are "**observed**", i.e. their values are given

Example:

$$P(X_2, X_5 \mid X_1 = 0, X_3 = 1, X_4 = 1)$$



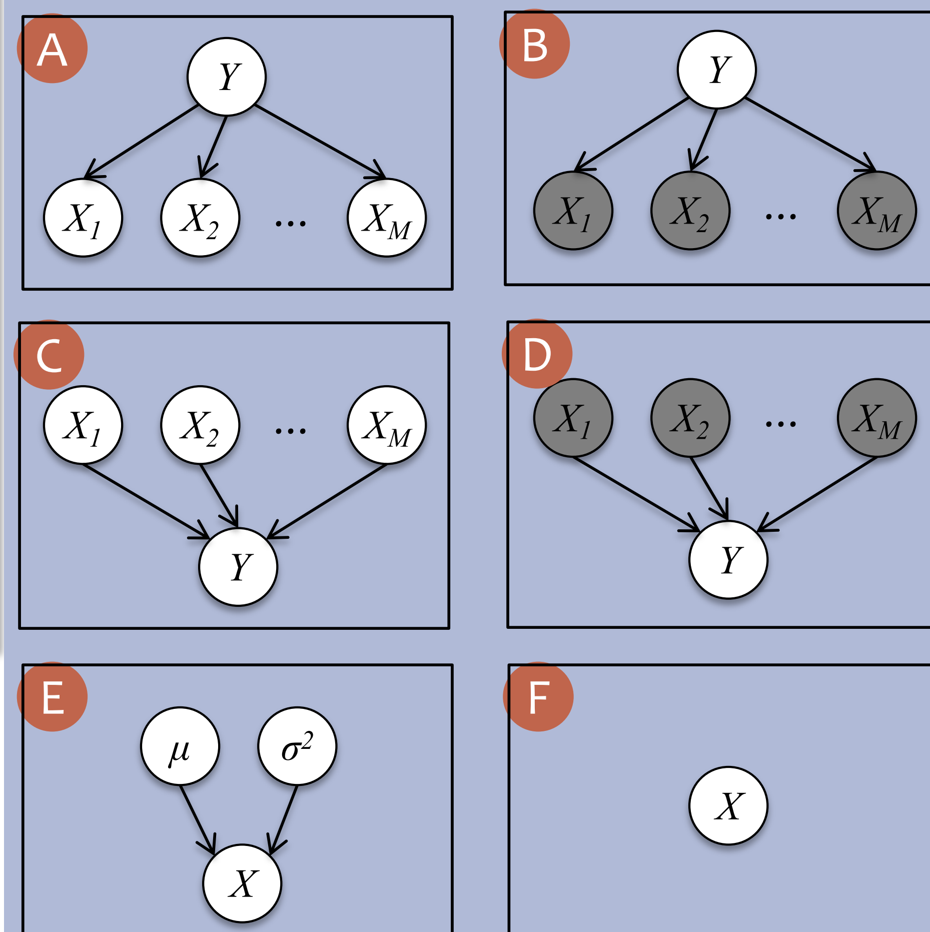
Familiar Models as Bayesian Networks

Question:

Match the model name to the corresponding Bayesian Network

1. Logistic Regression
2. Linear Regression
3. Bernoulli Naïve Bayes
4. Gaussian Naïve Bayes
5. 1D Gaussian

Answer:



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 $P(X_1, \ldots, X_T) = \prod_{t=1}^{T} P(X_t \mid \text{parents}(X_t))$

$$= \prod_{t=1}^{T} P(X_t \mid X_1, \ldots, X_{t-1})$$

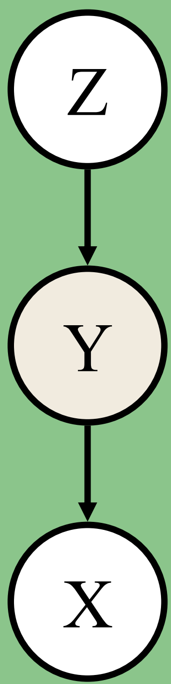
- But what else does it imply?

Slide from William Cohen

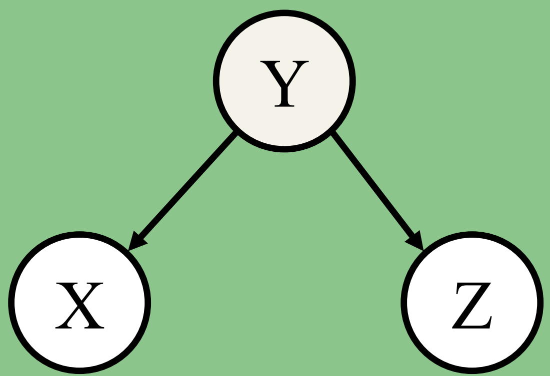
What Independencies does a Bayes Net Model?

Three cases of interest…

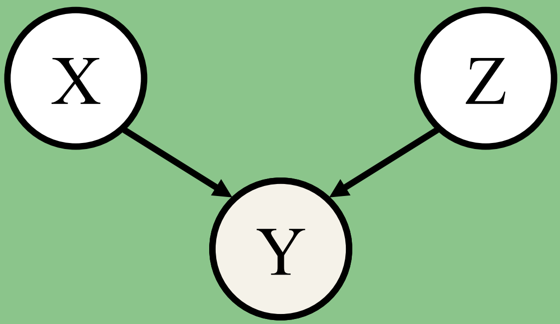
Cascade



Common Parent



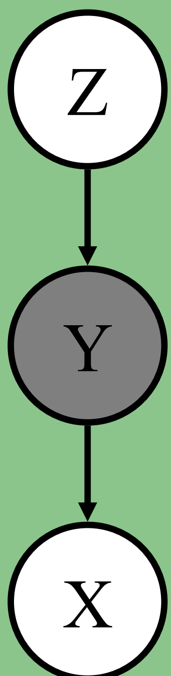
V-Structure



What Independencies does a Bayes Net Model?

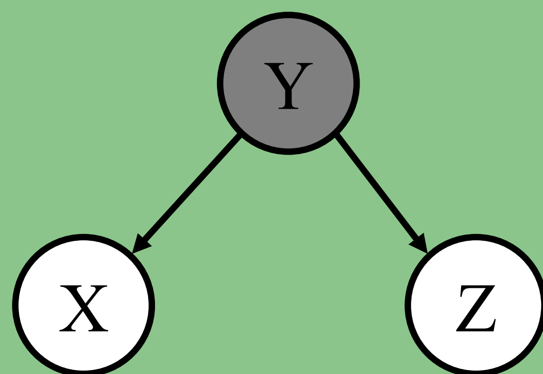
Three cases of interest...

Cascade



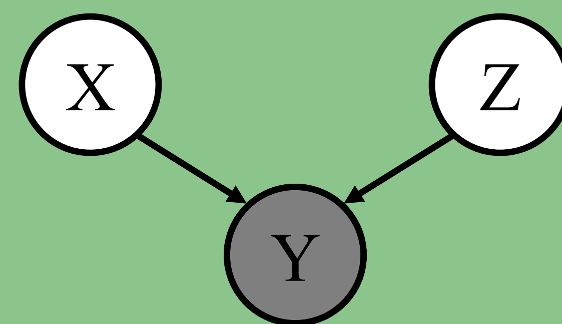
$$X \perp\!\!\!\perp Z \mid Y$$

Common Parent



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

V-Structure



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

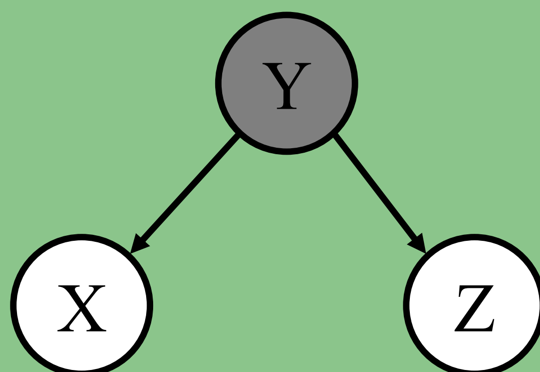
Knowing Y
decouples X and Z

Knowing Y
couples X and Z

Whiteboard

Proof of conditional independence

Common Parent



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

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