10-601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

HMMs

+

Bayesian Networks

Matt Gormley
Lecture 21
Apr. 01, 2020

Reminders

- Practice Problems for Exam 2
  - Out: Fri, Mar 20
- Midterm Exam 2
  - Thu, Apr 2 – evening exam, details announced on Piazza
- Homework 7: HMMs
  - Out: Thu, Apr 02
  - Due: Fri, Apr 10 at 11:59pm

- Today's In-Class Poll
  - http://poll.mlcourse.org

THE FORWARD-BACKWARD ALGORITHM

Forward-Backward Algorithm

<u>Define:</u> $\alpha_t(k) \triangleq p(x_1, \ldots, x_t, y_t = k)$

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

<u>Assume</u> $y_0 = \text{START}$

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

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

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

forward

② For $t = 1, \ldots, T$:

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

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

the alphas include the emission probabilities so we don't multiply them in separately

back

③ For $t = T, \ldots, T$:

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

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

O(K)

O(K²T)

④ Compute $p(\vec{x}) = \alpha_{T+1}(\text{END})$ [Evaluation]

⑤ Compute $p(y_t = k \mid \vec{x}) = \dfrac{\alpha_t(k)\, \beta_t(k)}{p(\vec{x})}$ [Marginals]

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

Inference for HMMs

Whiteboard

- Forward-backward algorithm (edge weights version)
- Viterbi algorithm (edge weights version)

Forward-Backward Algorithm

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

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

Assume $y_0 = \text{START}$

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

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

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

② For $t = 1, \ldots, T$:

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

the alphas include the emission probabilities so we don't multiply them in separately

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

O(K)

O(K²T)

③ For $t = T, \ldots, T$:

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 Compute $p(\vec{x}) = \alpha_{T+1}(\text{END})$ [Evaluation]

⑤ Compute $p(y_t = k \mid \vec{x}) = \dfrac{\alpha_t(k)\, \beta_t(k)}{p(\vec{x})}$ [Marginals]

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

Derivation of Forward Algorithm

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

Derivation:

Herein using "y_T" as shorthand for "$y_T = \text{END}$"

$$\begin{aligned}
\alpha_T(\text{END}) &= p(x_1, \ldots, x_T, y_T = \text{END}) \\
&= p(x_1, \ldots, x_T \mid y_T)\, p(y_T) && \leftarrow \text{by def. of joint} \\
&= p(x_T \mid y_T)\, p(x_1, \ldots, x_{T-1} \mid y_T)\, p(y_T) && \leftarrow \text{by cond. indep. of HMM} \\
&= p(x_T \mid y_T)\, p(x_1, \ldots, x_{T-1}, y_T) && \leftarrow \text{by def. of joint} \\
&= p(x_T \mid y_T) \sum_{y_{T-1}} p(x_1, \ldots, x_{T-1}, y_{T-1}, y_T) && \leftarrow \text{by def. of marginal} \\
&= p(x_T \mid y_T) \sum_{y_{T-1}} p(x_1, \ldots, x_{T-1}, y_T \mid y_{T-1})\, p(y_{T-1}) && \leftarrow \text{by def. of joint} \\
&= p(x_T \mid y_T) \sum_{y_{T-1}} p(x_1, \ldots, x_{T-1} \mid y_{T-1})\, p(y_T \mid y_{T-1})\, p(y_{T-1}) && \leftarrow \text{by cond. indep. of HMM} \\
&= p(x_T \mid y_T) \sum_{y_{T-1}} p(x_1, \ldots, x_{T-1}, y_{T-1})\, p(y_T \mid y_{T-1}) && \leftarrow \text{by def. of joint} \\
&= p(x_T \mid y_T) \sum_{y_{T-1}} \alpha_{T-1}(y_{T-1})\, p(y_T \mid y_{T-1}) && \leftarrow \text{by def. of } \alpha_t(k)
\end{aligned}$$

Viterbi Algorithm

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

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

Assume $y_0 = \text{START}$

① Initialize $\underbrace{\omega_0(\text{START})=1}$ $\underbrace{\omega_0(k)=0 \;\; \forall k \neq \text{START}}$

$O(K^2T)$

② For $t=1,\ldots,T$:

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

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

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

③ Compute $\boxed{\text{Most Probable Assignment}}$

[Decoding]

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

For $t=T-1,\ldots,1$

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

follow the "backpointers"

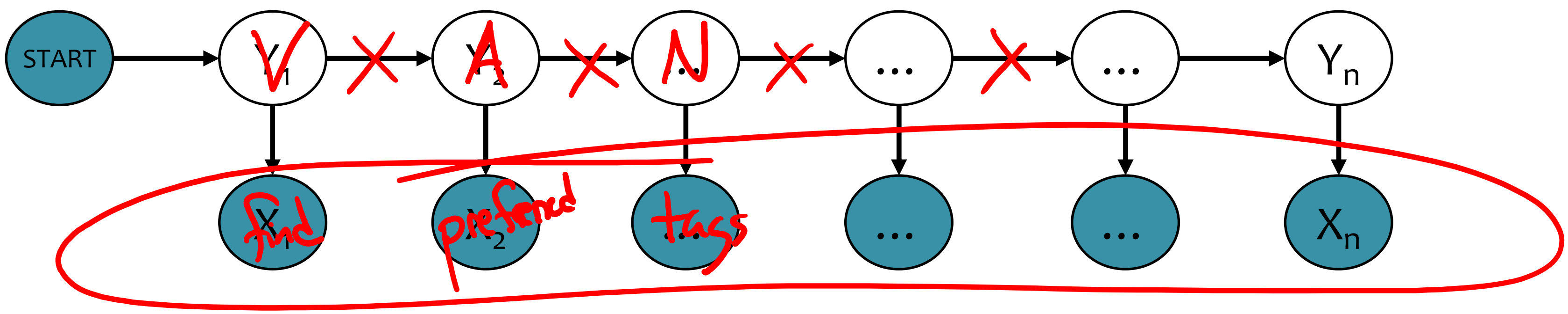
Inference in HMMs

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

– ~~Three~~ Four 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. F-B 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

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

$$= \underset{\hat{\boldsymbol{y}}}{\operatorname{argmin}} \sum_{\boldsymbol{y}} p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) \ell(\hat{\boldsymbol{y}}, \boldsymbol{y})$$

Minimum Bayes Risk Decoding

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

V A N → 0
V A N
A A A → 1

Consider some example loss functions:

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

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

The MBR decoder is:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \underset{\hat{\boldsymbol{y}}}{\text{argmin}} \sum_{\boldsymbol{y} \in \mathcal{Y}_x} p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x})(1 - \mathbb{I}(\hat{\boldsymbol{y}}, \boldsymbol{y}))$$

0 if $\hat{y} \neq y$
1 if $\hat{y} = y$

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

which is exactly the Viterbi decoding problem!

Minimum Bayes Risk Decoding

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

AAN
VAN
AAA
→ 1
→ 2

Consider some example loss functions:

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

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

marginal from F-B

The MBR decoder is:

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

This decomposes across variables and requires the variable marginals.

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

Bayes Nets Outline

- **Motivation**
  - Structured Prediction
- **Background**
  - Conditional Independence
  - Chain Rule of Probability
- **Directed Graphical Models**
  - Writing Joint Distributions
  - Definition: Bayesian Network
  - Qualitative Specification
  - Quantitative Specification
  - Familiar Models as Bayes Nets
- **Conditional Independence in Bayes Nets**
  - Three case studies
  - D-separation
  - Markov blanket
- **Learning**
  - Fully Observed Bayes Net
  - (Partially Observed Bayes Net)
- **Inference**
  - Background: Marginal Probability
  - Sampling directly from the joint distribution
  - Gibbs Sampling

Bayesian Networks

DIRECTED GRAPHICAL MODELS

Example: Ryan Reynolds' Voicemail

From https://www.adweek.com/brand-marketing/ryan-reynolds-left-voicemails-for-all-mint-mobile-subscribers/

Example: Ryan Reynolds Voicemail

Images from imdb.com

Example: Ryan Reynolds' Voicemail

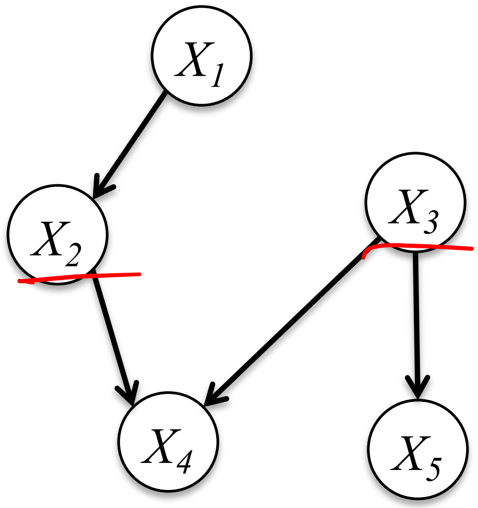
From https://www.adweek.com/brand-marketing/ryan-reynolds-left-voicemails-for-all-mint-mobile-subscribers/

Directed Graphical Models (Bayes Nets)

Whiteboard

- Example: Ryan Reynolds' Voicemail
- 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

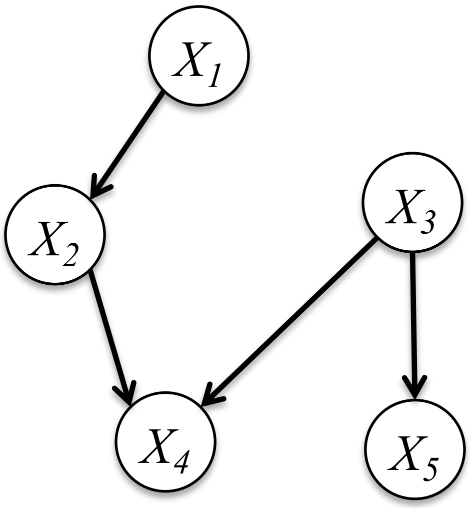
Bayesian Network



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

Bayesian Network

Definition:



$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid parents(X_i))$$

- 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

acyclic directed graph (DAG)

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

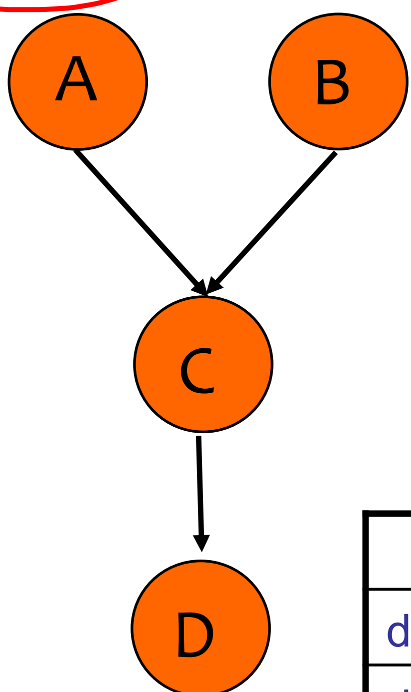
P(a)

a^0	0.75
a^1	0.25

P(b)

b^0	0.33
b^1	0.67

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



P(c|a,b)

	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

P(d|c)

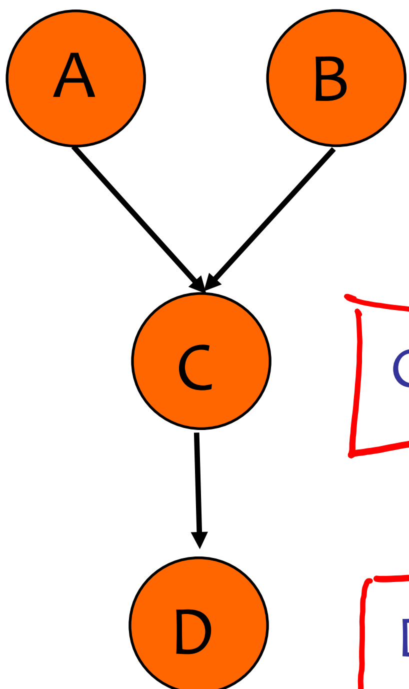
	c^0	c^1
d^0	0.3	0.5
d^1	07	0.5

Quantitative Specification

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

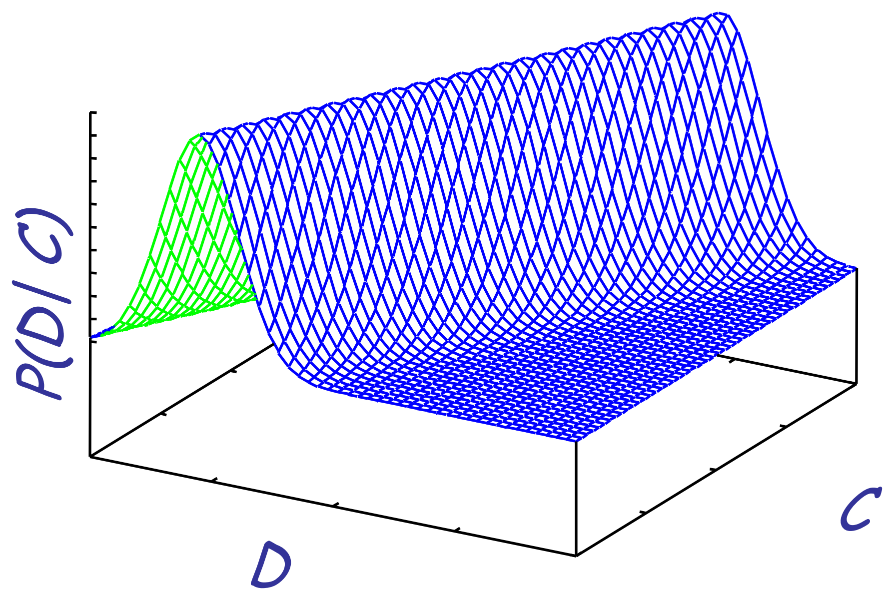
$A \sim N(\mu_a, \Sigma_a)$ $\quad$ $B \sim N(\mu_b, \Sigma_b)$

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



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

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



Quantitative Specification

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

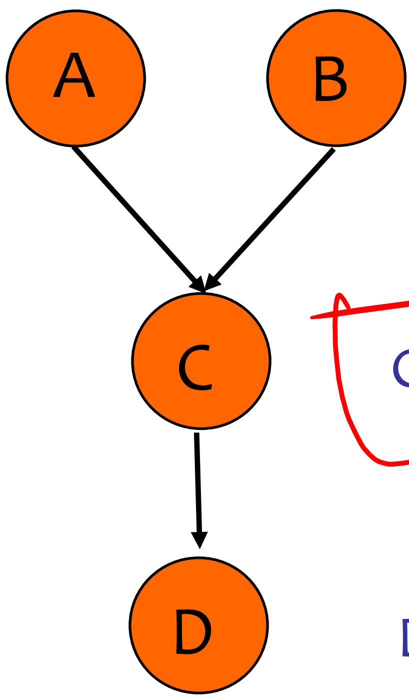
P(a)

a^0	0.75
a^1	0.25

P(b)

b^0	0.33
b^1	0.67

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



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

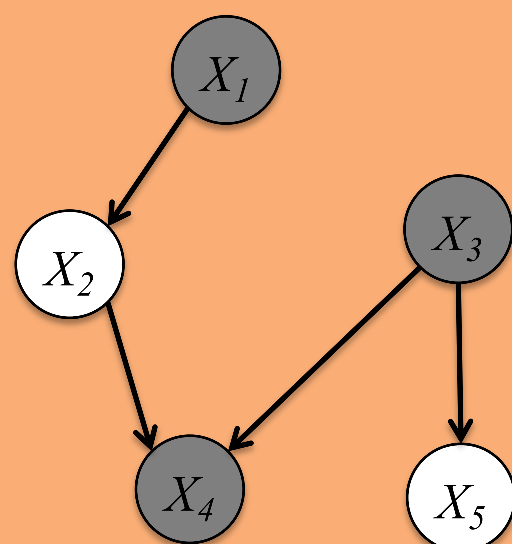
$D \sim N(\mu_d + C, \Sigma_d)$

Observed Variables

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

Example:

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



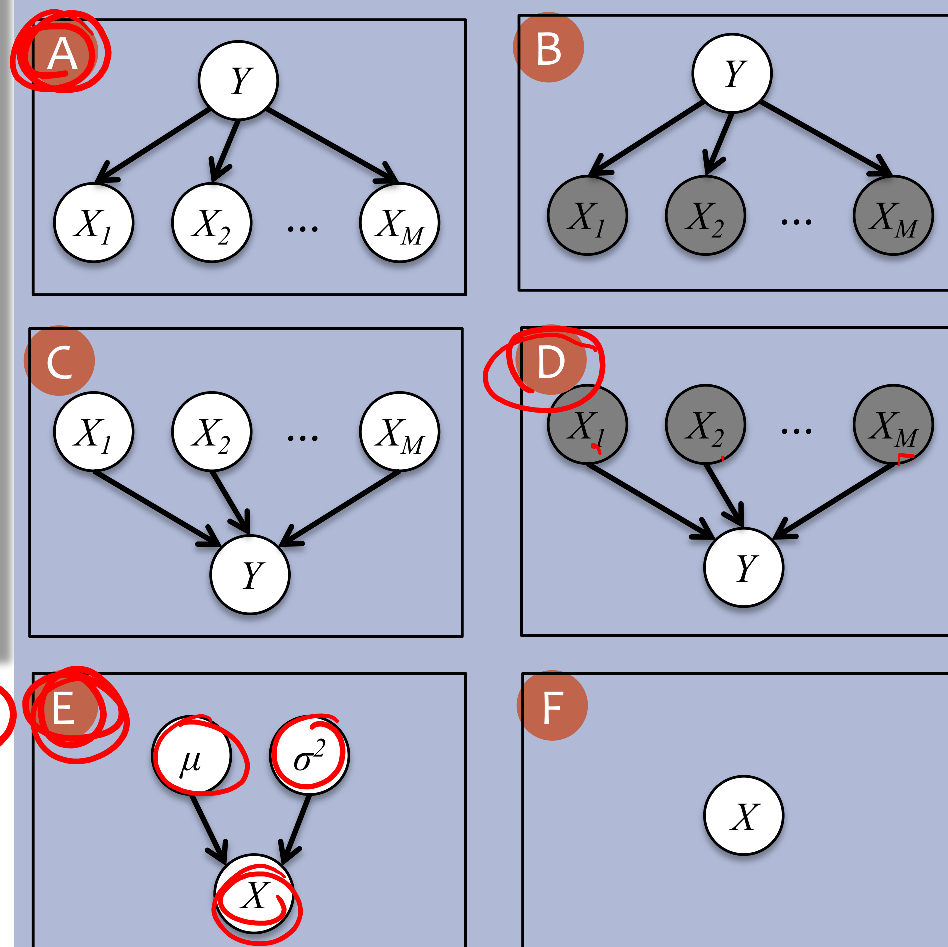
Familiar Models as Bayesian Networks

Question:

Match the model name to the corresponding Bayesian Network

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

Answer:



G = calamity

③ $p(y, x_1, \ldots, x_M) = p(y)\,p(x_1|y) \cdots p(x_M|y)$

④ →

① $p(y|x_1, \ldots, x_M)$

② →

⑤ $p(x|\mu, \sigma^2)$