

COMS W4701: Artificial Intelligence

Lecture 10: Bayesian Networks

Tony Dear, Ph.D.

Department of Computer Science

School of Engineering and Applied Sciences

Review: Inference in HMMs

- **Forward algorithm** for state estimation: Find $\mathbf{f}_t = P(X_t | e_{1:t})$

- Start from $\mathbf{f}_0 = P(X_0)$ and compute:

$$\begin{aligned} \mathbf{f}'_{t+1} &= T \mathbf{f}_t \\ \mathbf{f}_{t+1} &\propto_{X_{t+1}} O_{t+1} \mathbf{f}'_{t+1} \end{aligned}$$

$T_{ij} = P(X_{t+1} = i | X_t = j)$
 $(O_t)_{ii} = P(E_t = e_t | X_t = i)$

- **Viterbi algorithm** to find most likely sequence of states: $\operatorname{argmax}_{x_{1:t}} P(x_{1:t} | e_{1:t})$

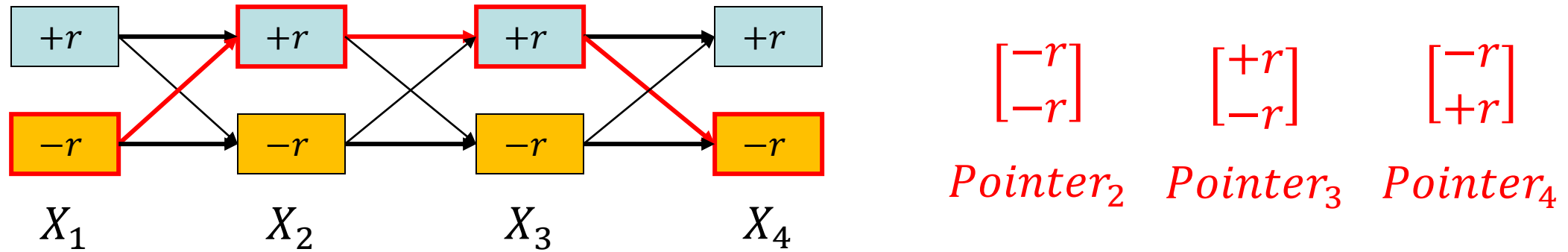
- Compute the joint distribution $\mathbf{m}_t = \max_{x_1 \dots x_{t-1}} P(x_{1:t-1}, X_t, e_{1:t})$

- Start from $\mathbf{m}_0 = P(X_0)$ and compute:

$$\begin{aligned} \mathbf{m}'_{t+1}(x_{t+1}) &= \max_{x_t} P(x_{t+1} | x_t) \mathbf{m}_t(x_t) \\ \mathbf{m}_{t+1} &= O_{t+1} \mathbf{m}'_{t+1} \end{aligned}$$

$$\mathbf{m}'_{t+1} = \begin{pmatrix} \max(T_{11}\mathbf{m}_t(1), T_{12}\mathbf{m}_t(2), \dots, T_{1n}\mathbf{m}_t(n)) \\ \max(T_{21}\mathbf{m}_t(1), T_{22}\mathbf{m}_t(2), \dots, T_{2n}\mathbf{m}_t(n)) \\ \vdots \\ \max(T_{n1}\mathbf{m}_t(1), T_{n2}\mathbf{m}_t(2), \dots, T_{nn}\mathbf{m}_t(n)) \end{pmatrix}$$

Review: Viterbi Algorithm



- Also need to keep track of most likely prior state when computing each \mathbf{m}_{t+1}

$$Pointer_{t+1}(x_{t+1}) = \underset{x_t}{\operatorname{argmax}} P(x_{t+1}|x_t) \mathbf{m}_t(x_t)$$

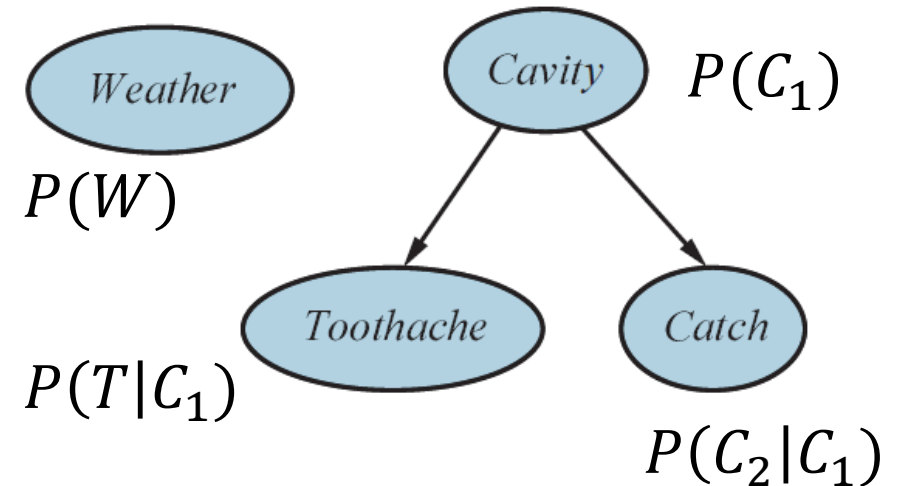
- $\underset{x_T}{\operatorname{argmax}} \mathbf{m}_T$ is the last state in most likely sequence
- Backward pass: Follow pointers backward to iteratively obtain each prior state

Today

- Last time: Hidden Markov models help us reason about belief states that evolve over space or time while incorporating observations
- This time: **Bayesian networks** are general graph structures that can represent arbitrary relationships and independences
- Naïve Bayes models and maximum likelihood learning

Bayesian Networks

- Recall: A set of RVs is described by their joint distribution
- **Bayesian network:** A directed acyclic graph (DAG) representation of a distribution
- Each node corresponds to a random variable
- Each edge indicates influence or correlation
- May also be causation, but not always
- **Parameters** of the Bayes net: A conditional probability table for each node
- The table for a node X_i contains the values $P(X_i | \text{parents}(X_i))$



Joint Distribution

- Recall that the *chain rule* gives us a general way to compute joint distributions:

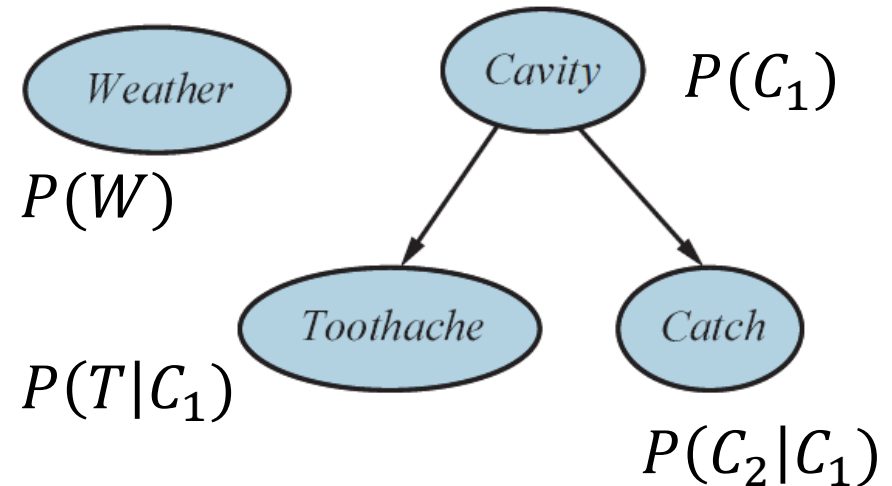
$$P(x_1, \dots, x_n) = P(x_1)P(x_2|x_1) \cdots P(x_n|x_1, \dots, x_{n-1}) = \prod_{i=1}^n P(x_i|x_1, \dots, x_{i-1})$$

- By definition, a joint distribution in a Bayes net can be written as

$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i|\text{parents}(X_i))$$

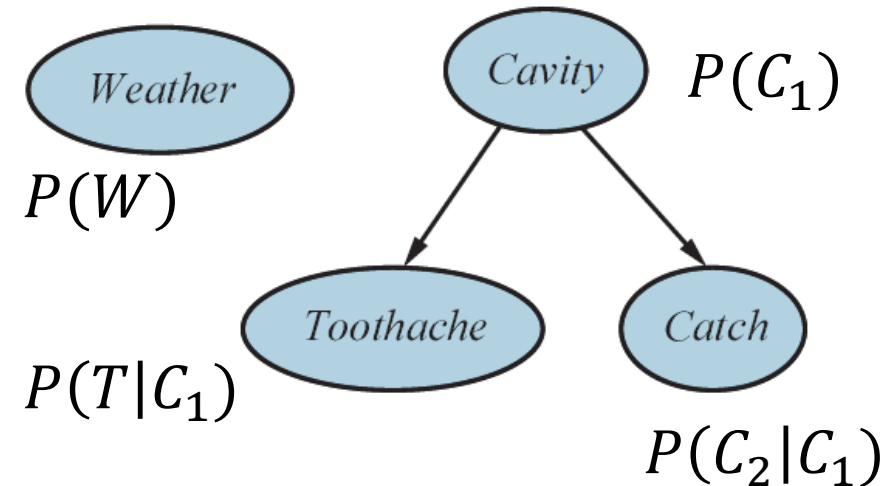
- This can simplify a lot of computation!

- Example:
$$P(w, c_1, t, c_2) = P(w)P(c_1)P(t|c_1)P(c_2|c_1)$$



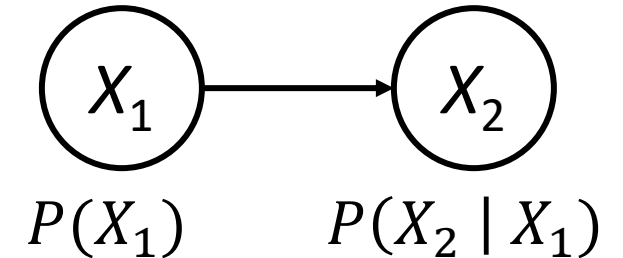
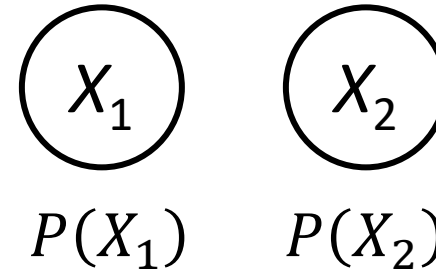
Topological Ordering

- We asserted that
$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i | \text{parents}(X_i))$$
- This implies that a node is conditionally independent of predecessors given its parents
- *Numbering* must correspond to **topological ordering**: parents before children
- One possible ordering: W, C_1, T, C_2
- Another one: C_1, C_2, T, W
- Another one: C_1, C_2, W, T
- Many more...



Constructing Bayes Nets

- Bayes net representations are not unique!
- Example: 2 (independent) coin flips
- Can use either $P(X_2)$ or $P(X_2|X_1)$
- Independence is deduced from probabilities
- No path *guarantees* independence between two RVs
- Presence of a path *may* indicate correlation between two RVs
- May also be independent, depending on probabilities
- Ex: $P(X_2|X_1)$ is storing redundant information



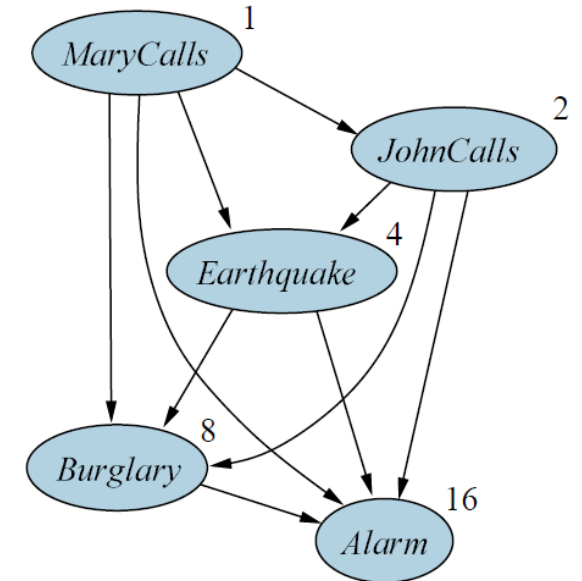
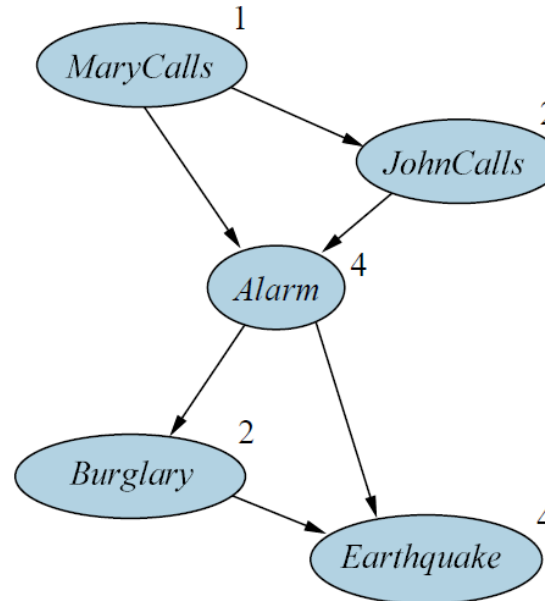
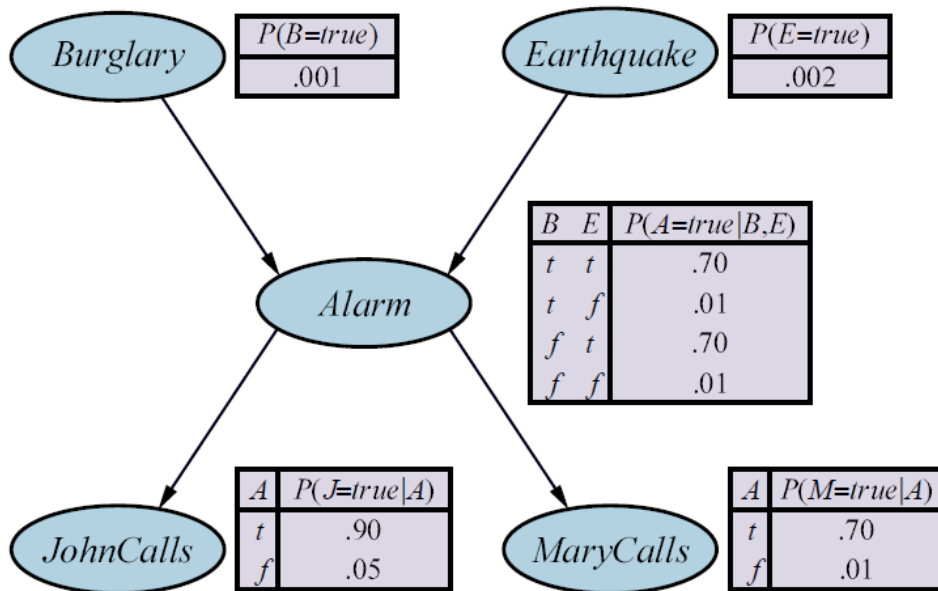
X_1	X_2	$P(X_2 X_1)$
+x	+x	0.5
+x	-x	0.5
-x	+x	0.5
-x	-x	0.5

Compactness

- When constructing a Bayes net, **compact** structures are more efficient
- The more independences that we can assert, the fewer the connections and parameters that we need to store
- Consider a distribution of n RVs, each with domain size d
- Full joint distribution is specified by a table of d^n numbers
- If any node has at most k parents, then its distribution is $O(d^{k+1})$
- We only need $O(n \times d^{k+1})$ parameters to store all information

Example: Alarm Network

- Boolean random variables: Each table has 2^k parameters
- All three structures can correctly represent same joint distribution
- Optimal structure has 10 parameters in total; others have 13 and 31

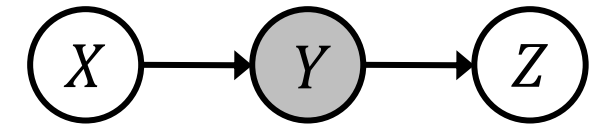
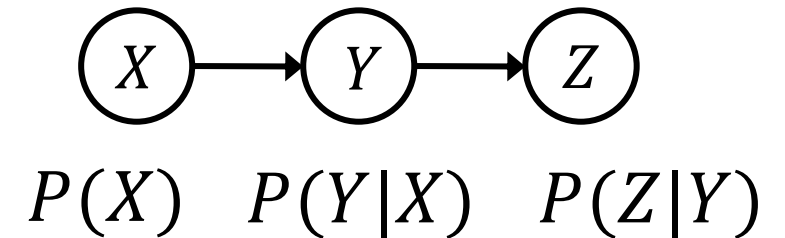


Conditional Independences

- Given a set of evidence variables, what conditional independences arise?
- We know that a node is independent of its “ancestors” given all its parents
- More generally, two nodes are conditionally independent if they are **d-separated** with *colliders* in all paths between them
- Here, “paths” do not depend on edge directionality!
- We identify three scenarios, which can generalize to any situation

Causal Chain

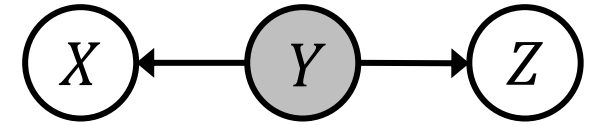
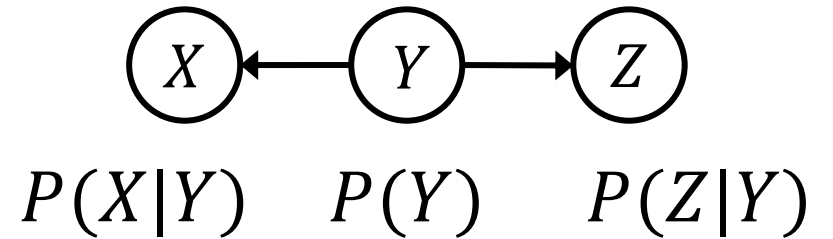
- Nodes in a *causal chain* (edges pointing in same direction) generally not independent
- But an observed node acts a *collider*
- Nodes on one side become independent of nodes on the other



- Proof:
$$P(x, z|y) = \frac{P(x)P(y|x)P(z|y)}{P(y)}$$
$$= \frac{P(x)P(z|y)}{P(y)} \frac{P(x|y)P(y)}{P(x)} = P(z|y)P(x|y)$$

Common Cause

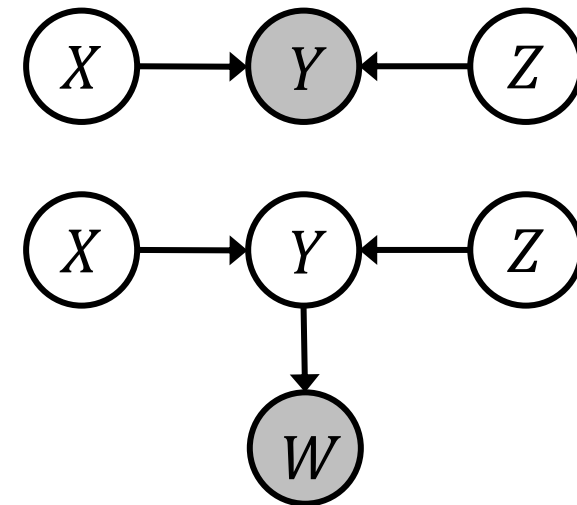
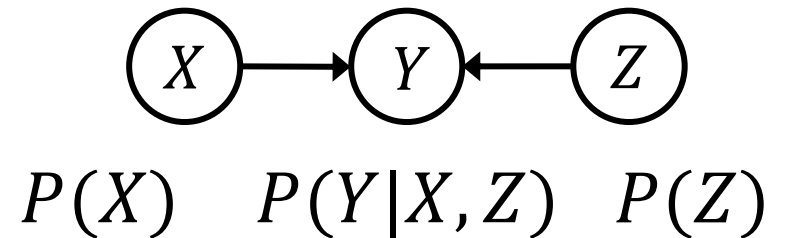
- Nodes connected by a *common cause* (arc heads pointing away from each other) are generally not independent
- But an observed node cause acts a *collider*
- Nodes on one side become independent of nodes on the other



- **Proof:**
$$P(x, z|y) = \frac{P(x, y, z)}{P(y)} = \frac{P(y)P(x|y)P(z|y)}{P(y)} = P(x|y)P(z|y)$$

Common Effect

- A *common effect* (arc heads pointing toward each other) between two nodes acts a *collider*
- Nodes on either side are independent
- But observing the effect or any of its descendants removes the collider!
- Nodes are all connected; independence no longer guaranteed



Example: Common Effect

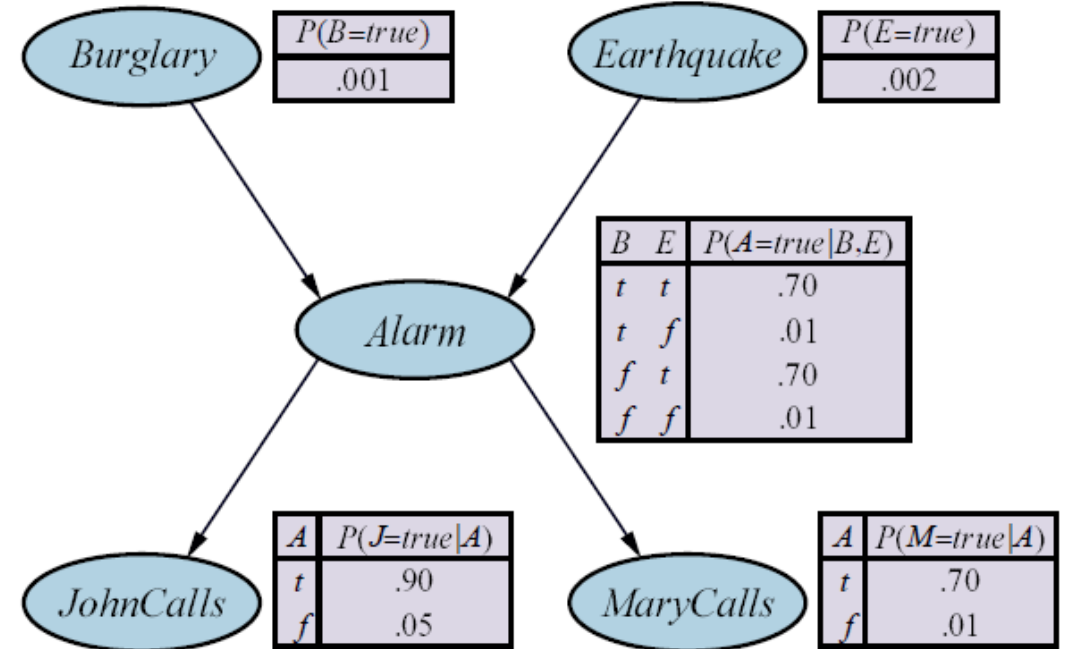
$$P(+b, +e) = P(+b)P(+e) = 0.001 \times 0.002 = 2 \times 10^{-6}$$

$$P(+b, +e | +a) = \frac{P(+b, +e, +a)}{P(+a)}$$

$$= \frac{P(+b)P(+e)P(+a | +b, +e)}{\sum_{b,e} P(b)P(e)P(+a | b, e)}$$

$$= \frac{0.001 \times 0.002 \times 0.7}{.001(.002)(.7) + .001(.998)(.01) + .999(.002)(.7) + .999(.998)(.01)}$$

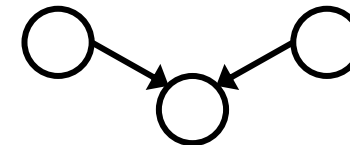
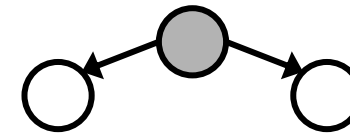
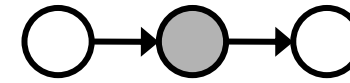
$$= 1.23 \times 10^{-4}$$



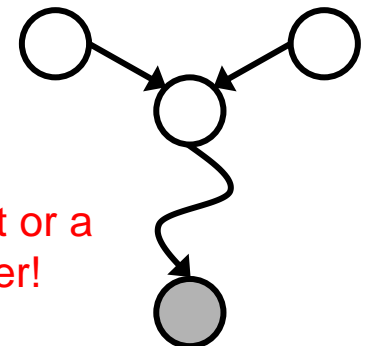
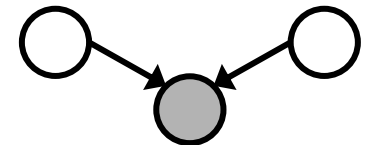
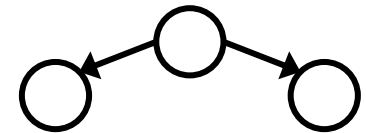
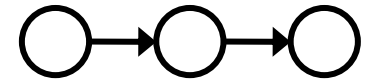
D-Separation

- Question: Are two nodes X_i and X_j independent conditioned on a set of nodes E ?
- First identify “conditioned” nodes (shade them in)
- If *at least one path* between X_i and X_j is unblocked (no colliders): not conditionally independent
- If *every path* between X_i and X_j contains a collider: conditionally independent!

Collider



No collider

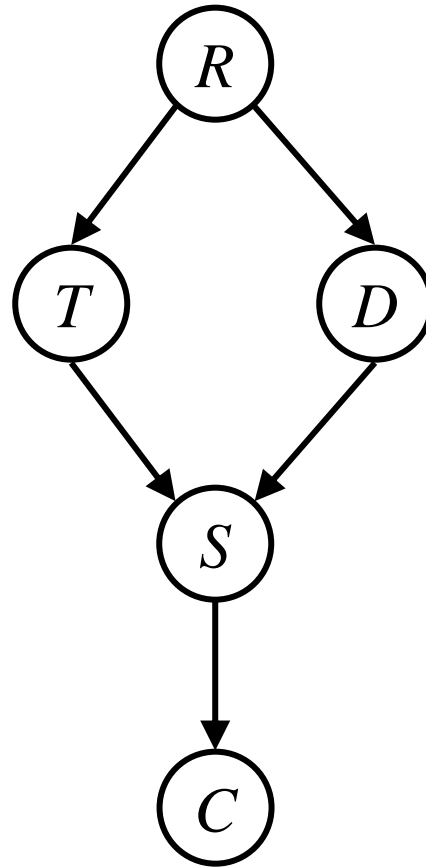


For the common effect case, conditioning on either the effect or a descendant removes the collider!

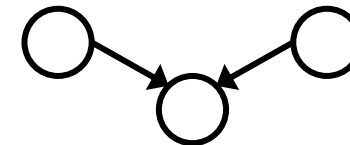
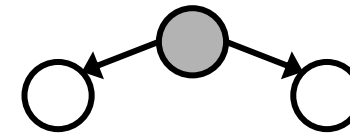
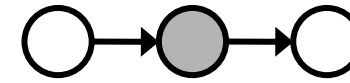
Example: D-Separation

Which nodes are independent...

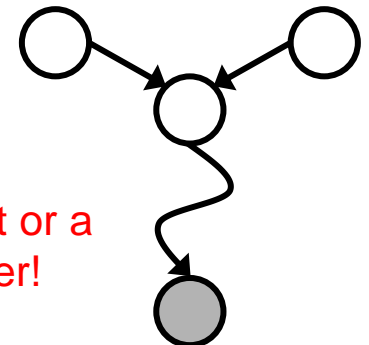
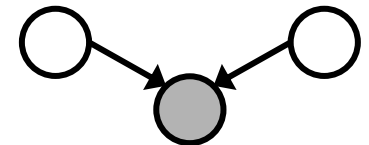
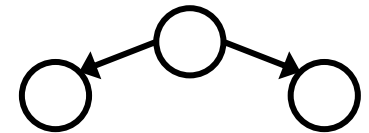
- Given S ?
- Given R ?
- Given T or D ?
- Given T and D ?
- Given R and S ?
- Given R and C ?



Collider



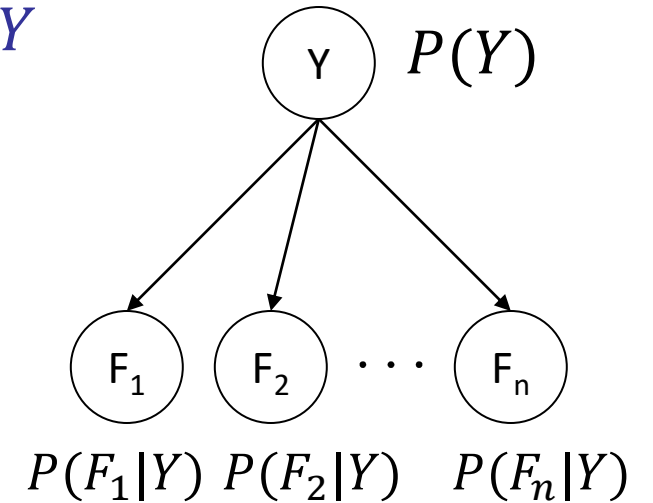
No collider



For the common effect case,
conditioning on either the effect or a
descendant removes the collider!

Naïve Bayes Models

- In a naïve Bayes model, there is an underlying “cause” variable Y
- Y then influences a number of different effects F_i
- Given Y , all the F_i are conditionally independent!
- Joint distribution: $P(y, f_1, \dots, f_n) = P(y) \prod_{i=1}^n P(f_i|y)$
- How to *infer* the cause given observed effects?



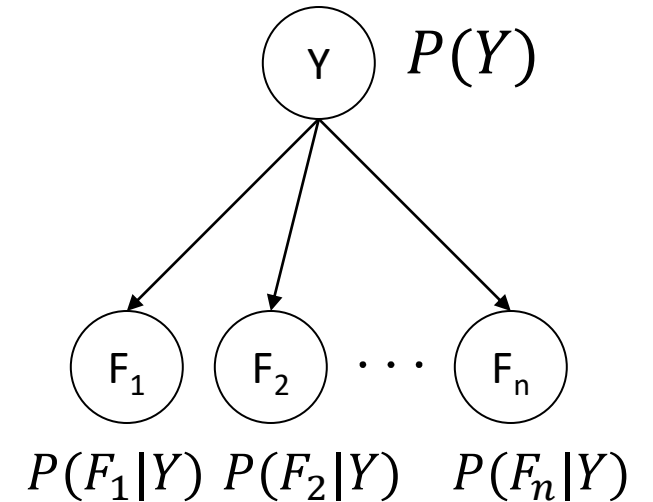
$$P(Y|f_1, \dots, f_n) = \frac{P(Y, f_1, \dots, f_n)}{P(f_1, \dots, f_n)} \propto_Y P(Y) \prod_{i=1}^n P(f_i|Y)$$

Classification

- Naïve Bayes can be used for **classification** tasks
- Given a data input, predict an output *label* or *class*
- Inputs are *features* f_i and output is most likely class y
- Predicted class:

$$y = \operatorname{argmax}_y P(y|f_1, \dots, f_n) = \operatorname{argmax}_y P(y) \prod_{i=1}^n P(f_i|y)$$

- We don't even need to normalize—exact probabilities are not required
- Problem: If we have too many features, product will underflow



Log Probabilities

- As with Viterbi, NB classification only looks at relative ordering of likelihoods, not the actual probabilities
- *Logarithm* function preserves ordering and can address underflow issues

$$\operatorname{argmax}_y P(y) \prod_{i=1}^n P(f_i|y) = \operatorname{argmax}_y \log \left(P(y) \prod_{i=1}^n P(f_i|y) \right)$$

- Log of a product = sum of logs

$$\log \left(P(y) \prod_{i=1}^n P(f_i|y) \right) = \log(P(y)) + \sum_{i=1}^n \log(P(f_i|y))$$

- Since probabilities ≤ 1 , log probabilities are always ≤ 0

Classification Examples

- Spam filter: Inputs are emails, class/label is spam/ham
- Emails contain **features** that will help classify
 - Words: “wire money”, “bank account”, “supplements”
 - Text patterns: all caps, unusual font, dollar signs
 - Other flags: Sender not in contacts, geographic origin
- Digit recognition: Inputs are pixels, class/label is a digit
- Pixel grids contain **features** that will help classify
 - Whether specific pixels are ON or OFF
 - Patterns: Size, number of loops, number of lines



0



1



2



1



??

Example: Spam Filtering

- Labels $Y = \{\text{spam}, \text{ham}\}$
- Features W_i for position i in the email over common dictionary words
- “**Bag-of-words**” model: Each W_i is independent, identically distributed
 - No dependence on ordering or other nearby words!

Word	$P(Y)$	Hey	would	you	like	to	lose	weight
$P(w \text{spam})$	0.33333	0.00002	0.00069	0.00881	0.00086	0.01517	0.00008	0.00016
$P(w \text{ham})$	0.66666	0.00021	0.00084	0.00304	0.00083	0.01339	0.00002	0.00002

- $\log P(\text{spam}) + \sum \log P(w|\text{spam}) = -53.3$
- $\log P(\text{ham}) + \sum \log P(w|\text{ham}) = -55.0$

Supervised Learning

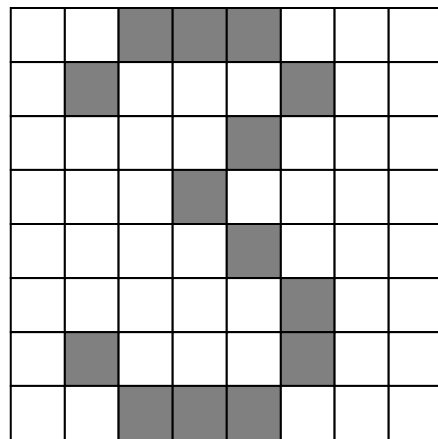
- Where do the probabilities (*parameters*) of our models come from?
- **Supervised learning:** Given *training data* with input-output pairs $(x_1, y_1), \dots, (x_N, y_N)$ generated by an unknown function f , find a *hypothesis* to best approximate f
- Expressiveness-complexity tradeoff: The more expressive the hypothesis space, the higher the computational complexity of searching for a good hypothesis
- Bias-variance tradeoff: More complex hypotheses that fit training data well, or simpler hypotheses that generalize to new data better?

Learning Naïve Bayes Models

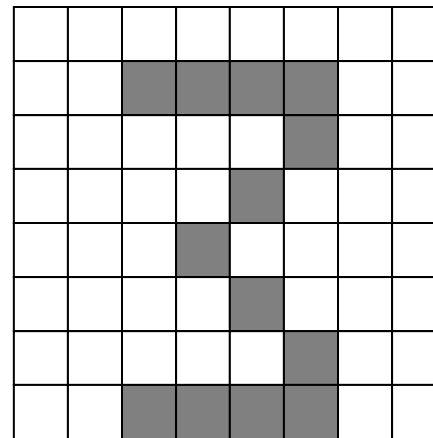
- If our data are identical and independently distributed (i.i.d.), the likelihood of all of them occurring given a hypothesis h is $P(data|h) = \prod_j P(data_j|h)$
- **Maximum likelihood learning:** Find the hypothesis that maximizes $P(data|h)$
- More specifically, find the hypothesis *parameters* θ that maximize $L(\theta) = P(data|h_\theta)$
- For Naïve Bayes, the parameters are the individual model probabilities $P(y), P(f_i|y)$
- $L(\theta)$ is maximized when the probabilities are chosen by simply *counting the occurrences of each event* in the data

Example: Digit Recognition

- Labels $Y = \{0,1,2,3,4,5,6,7,8,9\}$
- Binary features F_{ij} for each pixel (i,j) in the input image
- Suppose we have two training examples of 3s



$$\begin{aligned}
 Y &= 3 \\
 F_{00} &= 0 \\
 F_{01} &= 0 \\
 F_{02} &= 1 \\
 F_{03} &= 1 \\
 &\vdots \\
 F_{77} &= 0
 \end{aligned}$$



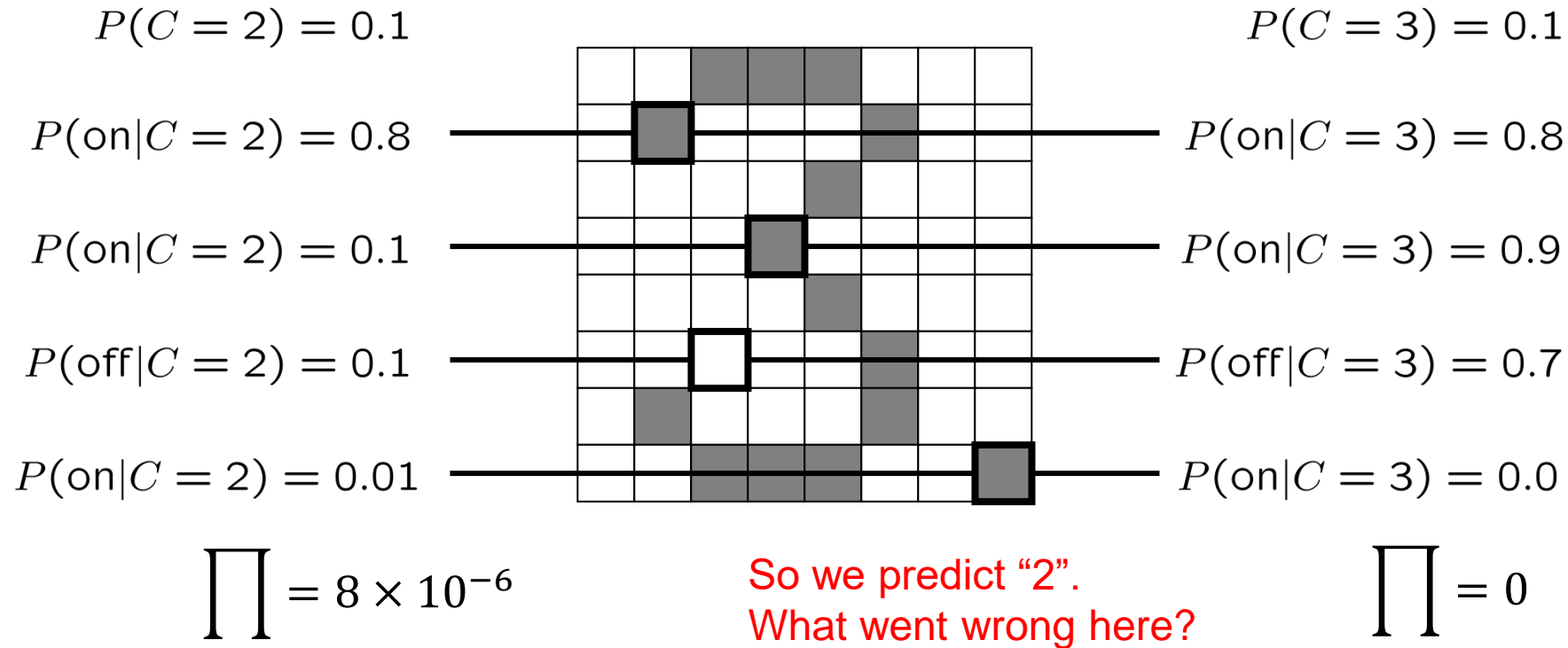
$$\begin{aligned}
 Y &= 3 \\
 F_{00} &= 0 \\
 &\vdots \\
 F_{12} &= 1 \\
 F_{13} &= 1 \\
 &\vdots \\
 F_{77} &= 0
 \end{aligned}$$

$$\begin{aligned}
 P(F_{00} = 1|Y = 3) &= 0 \\
 P(F_{01} = 1|Y = 3) &= 0 \\
 P(F_{02} = 1|Y = 3) &= 0.5 \\
 &\vdots \\
 P(F_{72} = 1|Y = 3) &= 1 \\
 &\vdots \\
 P(F_{77} = 1|Y = 3) &= 0
 \end{aligned}$$

- In practice, we will be using much more than just 2 training samples

Overfitting: Digit Recognition

- Suppose we learned a model to classify 2 or 3, but our training data has no 3s with the bottom-right pixel on



Overfitting: Spam Detection

- We will not see every dictionary word in training data—0s all over the place in the model!
- If any word appears “ham” data but never in “spam”, then any email with that word will be classified as “ham” (and vice-versa)
- “Dear Sir/Madam, I would like to **seriously** offer you an exclusive opportunity to win \$1M by simply sending us your bank info” -> HAM

$P(W|\text{spam})$



south-west	:	0
nation	:	0
morally	:	0
nicely	:	0
extent	:	0
seriously	:	0
...		

$P(W|\text{ham})$

screens	:	0
minute	:	0
guaranteed	:	0
\$205.00	:	0
delivery	:	0
signature	:	0
...		

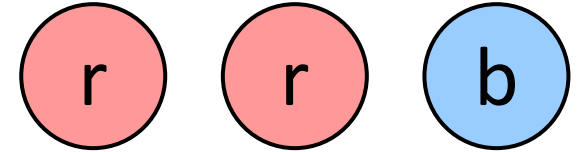
Additive Smoothing

- Problem : If a specific class/feature does not appear in training data, probability assumed to be 0!
 - This will automatically zero out $P(Y, F_1, \dots, F_n) = P(Y) \prod_i P(F_i|Y)$
 - To avoid overfitting, we need to **generalize** our models
- Idea: **Smooth** our estimates by adding a “pseudocount” k

$\hat{\theta} = \frac{c(x) + k}{N + k X }$	$k \rightarrow 0$ 	$\hat{\theta} = \frac{c(x)}{N}$	Original maximum likelihood estimator (average)
	$k \rightarrow \infty$ 	$\hat{\theta} = \frac{1}{ X }$	Uniform prior over all possibilities of X (ignore all samples)

Example: Additive Smoothing

- Estimate probabilities of red, blue, and green marbles




Color	$\hat{P}_{MLE}(\text{color})$ $= \hat{P}_{k0}(\text{color})$
Red	0.667
Blue	0.333
Green	0


Color	$\hat{P}_{k1}(\text{color})$
Red	$(2+1)/(3+3)=0.5$
Blue	$(1+1)/(3+3)=0.333$
Green	$(0+1)/(3+3)=0.167$

Color	$\hat{P}_{k100}(\text{color})$
Red	$(2+100)/(3+300)=0.337$
Blue	$(1+100)/(3+300)=0.333$
Green	$(0+100)/(3+300)=0.330$

$$\hat{\theta} = \frac{c(x) + k}{N + k|X|}$$

$k \rightarrow 0$


$\hat{\theta} = \frac{c(x)}{N}$

$k \rightarrow \infty$


$\hat{\theta} = \frac{1}{|X|}$

Original maximum likelihood estimator (average)

Uniform prior over all possibilities of X (ignore all samples)

Summary

- Bayesian networks graphically encode independence assumptions about joint distributions in a compact way
- D-separation rules can help infer additional independences given evidence
- Naïve Bayes models are simple ways of thinking about cause-effect or class-label relationships
- Supervised learning of model parameters can be done using maximum likelihood learning (for Naïve Bayes, simple counting)