STEVENS
INSTITUTE OF TECHNOLOGY
1870

Artificial Intelligence

Computer Science, CS541 - A

Jonggi Hong

Announcements

- HW #2: Logic
  - Due 11:59 pm, Tuesday, March 7
  - **Today!**

- Spring break
  - No class, no office hour

STEVENS
INSTITUTE OF TECHNOLOGY
1870

Recap

Bayesian Network

Bayesian Network (Bayes' Net)

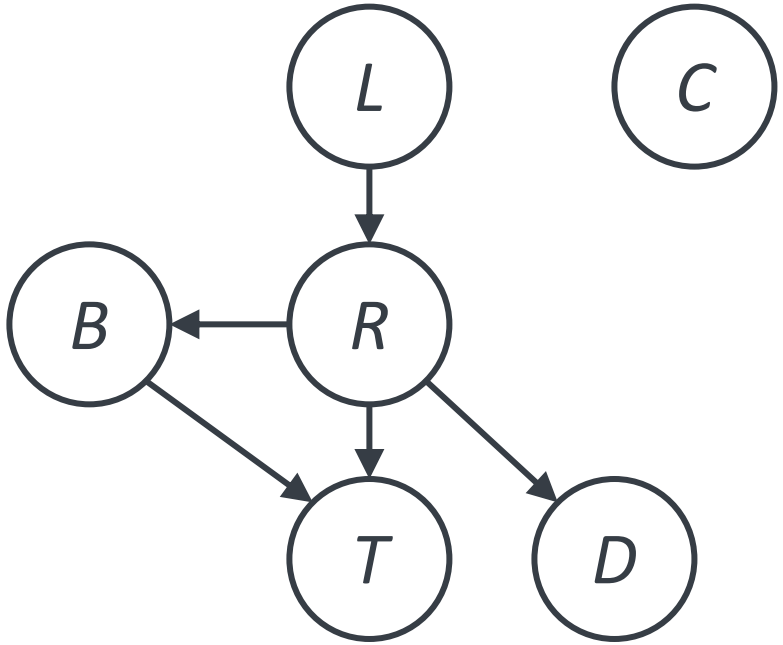
Bayes' Nets: Big Picture

- Two problems with using full joint distribution tables as our probabilistic models:
  - Unless there are only a few variables, the joint is WAY too big to represent explicitly
  - Hard to learn (estimate) anything empirically about more than a few variables at a time

- **Bayesian network**: a technique for describing complex joint distributions (models) using simple, local distributions (conditional probabilities)
  - More properly called graphical models
  - We describe how variables locally interact
  - Local interactions chain together to give global, indirect interactions

Example: Traffic II

- Let’s build a causal graphical model!
- Variables
  - T: Traffic
  - R: It rains
  - L: Low pressure
  - D: Roof drips
  - B: Ballgame
  - C: Cavity

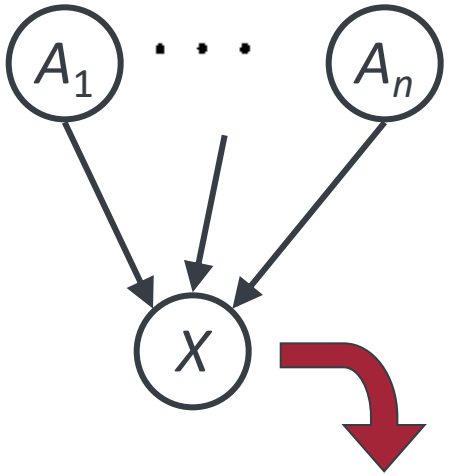


Bayesian Network Semantics

- A set of nodes, one per variable X
- A directed, acyclic graph
- A conditional distribution for each node
  - A collection of distributions over X, one for each combination of parents' values

$$P(X|a_1 \ldots a_n)$$

  - CPT: conditional probability table
  - Description of a noisy "causal" process



$$P(X|A_1 \ldots A_n)$$

A Bayes net = Topology (graph) + Local Conditional Probabilities

Probabilities in BNs

- Why are we guaranteed that this equation results in a proper joint distribution?

$$P(x_1, x_2, \ldots x_n) = \prod_{i=1}^{n} P(x_i | parents(X_i))$$

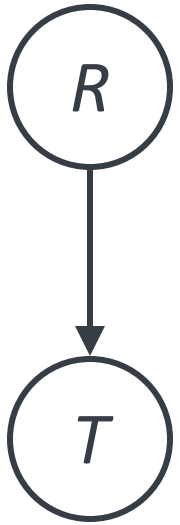
- Chain rule (valid for all distributions): $P(x_1, x_2, \ldots x_n) = \prod_{i=1}^{n} P(x_i | x_1 \ldots x_{i-1})$

- Assume conditional independences: $P(x_i | x_1, \ldots x_{i-1}) = P(x_i | parents(X_i))$

→ Consequence: $P(x_1, x_2, \ldots x_n) = \prod_{i=1}^{n} P(x_i | parents(X_i))$

- Not every BN can represent every joint distribution.
  - The topology enforces certain conditional independencies.

Example: Traffic



P(R)

+r	1/4
-r	3/4

P(T|R)

+r	+t	3/4
	-t	1/4
-r	+t	1/2
	-t	1/2

P(+r, -t) =

Causality?

- When Bayes' nets reflect the true causal patterns:
  - Often simpler (nodes have fewer parents)
  - Often easier to think about
  - Often easier to elicit from experts

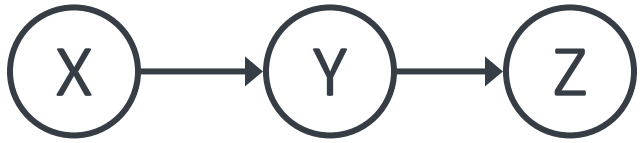
- BNs need not actually be causal
  - Sometimes no causal net exists over the domain (especially if variables are missing)
  - E.g., consider the variables *Traffic* and *Drips*
  - End up with arrows that reflect correlation, not causation

- What do the arrows really mean?
  - Topology may happen to encode causal structure.
  - **Topology really encodes conditional independence.**

$$P(x_i|x_1, \ldots x_{i-1}) = P(x_i|\mathit{parents}(X_i))$$

Independence in a BN

- Important question about a BN:
  - Are two nodes independent given certain evidence?
  - If yes, can prove using algebra (tedious in general)
  - If no, can prove with a counter example
  - Example:



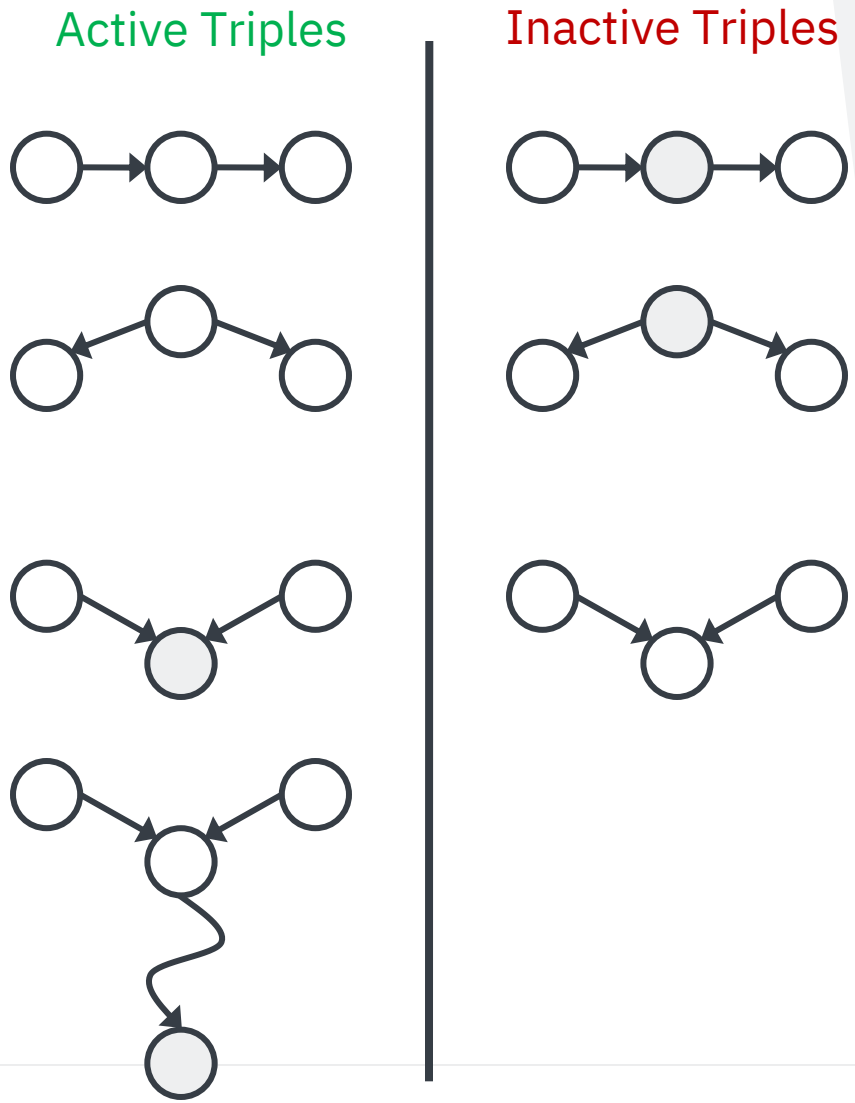
  - Question: are X and Z necessarily independent?
    - Answer: no. (Example: low pressure causes rain, which causes traffic.)
    - X can influence Z, Z can influence X (via Y)
    - Addendum: they *could* be independent: how?

D-separation

Active / Inactive Paths

- Question: Are X and Y conditionally independent given evidence variables {Z}?
  - Yes, if X and Y "d-separated" by Z
  - Consider all (undirected) paths from X to Y
  - No active paths = independence!
- A path is active if each triple is active:
  - Causal chain $A \rightarrow B \rightarrow C$ where B is unobserved (either direction)
  - Common cause $A \leftarrow B \rightarrow C$ where B is unobserved
  - Common effect (aka v-structure)

    $A \rightarrow B \leftarrow C$ where B *or one of its descendents* is observed
- All it takes to block a path is a single inactive segment.



D-Separation

- Query: $X_i \perp\!\!\!\perp X_j | \{X_{k_1}, ..., X_{k_n}\}$?

- Check all (undirected!) paths between X_i and X_j

  - If one or more active, then independence not guaranteed.

    $$X_i \not\perp\!\!\!\perp X_j | \{X_{k_1}, ..., X_{k_n}\}$$

  - Otherwise (*i.e.*, if all paths are inactive), then independence is guaranteed.

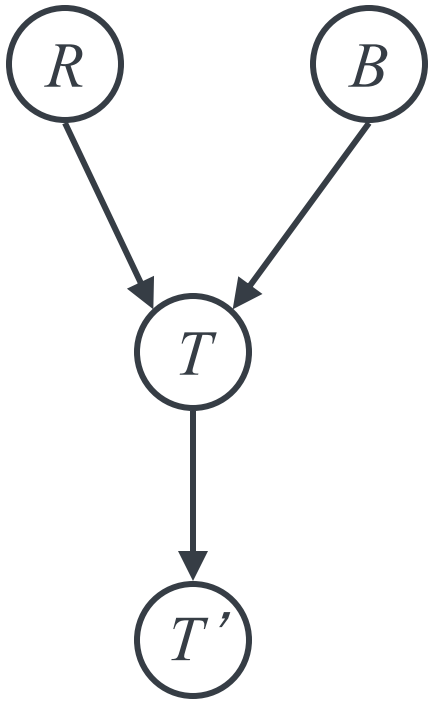
    $$X_i \perp\!\!\!\perp X_j | \{X_{k_1}, ..., X_{k_n}\}$$

Example

$R \perp\!\!\!\perp B$ *Yes*

$R \perp\!\!\!\perp B | T$

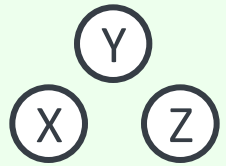
$R \perp\!\!\!\perp B | T'$

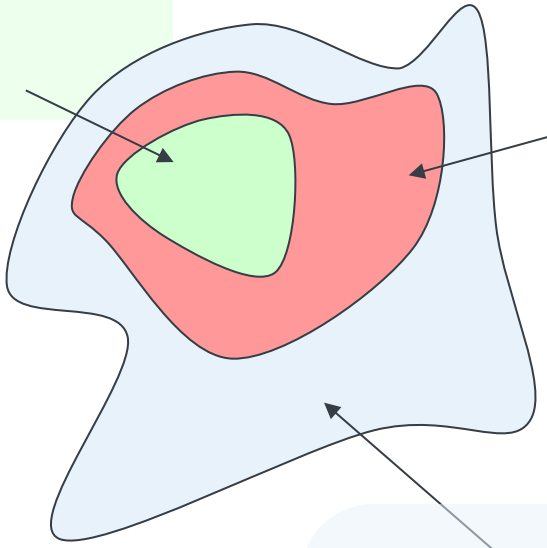


Topology Limits Distributions

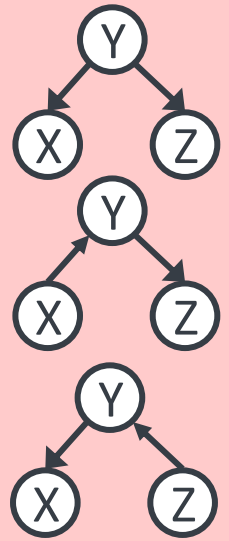
- Given some graph topology G, only certain joint distributions can be encoded.
- The graph structure guarantees certain (conditional) independences.
- (There might be more independence)
- Adding arcs increases the set of distributions, but has several costs.
- Full conditioning can encode any distribution.

$$\{X \perp\!\!\!\perp Y, X \perp\!\!\!\perp Z, Y \perp\!\!\!\perp Z, X \perp\!\!\!\perp Z \mid Y, X \perp\!\!\!\perp Y \mid Z, Y \perp\!\!\!\perp Z \mid X\}$$

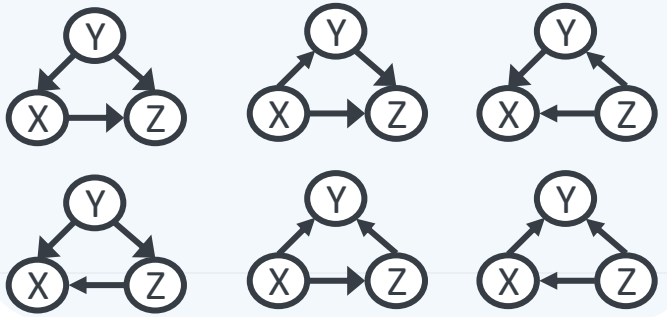




$$\{X \perp\!\!\!\perp Z \mid Y\}$$



$$\{\}$$

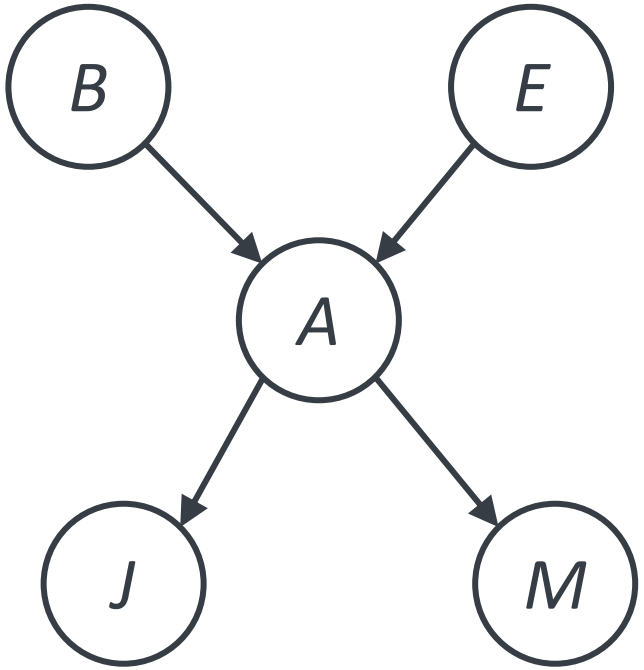


Inference

Inference

- What is P(B|+j, +m)?

$$P(Q|e_1 \dots e_k)$$



Inference

- Inference: calculating some useful quantity from a joint probability distribution

- Examples
  - Posterior priority $P(Q|E_1 = e_1, \ldots E_k = e_k)$
  - Most likely explanation $\text{argmax}_q \; P(Q = q|E_1 = e_1 \ldots)$

Inference by Enumeration

- General case:
  - Evidence variables: $E_1 \dots E_k = e_1 \dots e_k$
  - Query* variable: Q
  - Hidden variables: $H_1 \dots H_r$

$\left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} X_1, X_2, \dots X_n$

All variables

- We want: $P(Q|e_1 \dots e_k)$

** Works fine with multiple query variables, too*

- Step 1: Select the entries consistent with the evidence
- Step 2: Sum out H to get joint of Query and evidence
- Step 3: Normalize

$$P(Q, e_1 \dots e_k) = \sum_{h_1 \dots h_r} \underbrace{P(Q, h_1 \dots h_r, e_1 \dots e_k)}_{X_1, X_2, \dots X_n}$$

$$\times \frac{1}{Z}$$

$$Z = \sum_q P(Q, e_1 \cdots e_k)$$

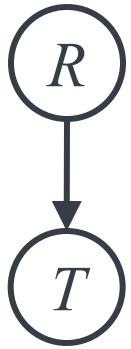
$$P(Q|e_1 \cdots e_k) = \frac{1}{Z} P(Q, e_1 \cdots e_k)$$

Inference by Enumeration vs. Variable Elimination

- Why is inference by enumeration so slow?
  - You join up the whole joint distribution before you sum out the hidden variables.
- Idea: interleave joining and marginalizing!
  - Called “Variable Elimination”
  - Still NP-hard, but usually much faster than inference by enumeration.

Operation 1: Join Factors

- First basic operation: **joining factors**
- Combining factors:
  - Just like a database join
  - Get all factors over the joining variable
  - Build a new factor over the union of the variables involved
- Example: Join on R
  - Computation for each entry: pointwise products



$$P(R) \times P(T|R) \rightarrow P(R,T)$$

+r	0.1
-r	0.9

+r	+t	0.8
+r	-t	0.2
-r	+t	0.1
-r	-t	0.9

+r	+t	0.08
+r	-t	0.02
-r	+t	0.09
-r	-t	0.81

$$\forall r, t: \quad P(r,t) = P(r) \cdot P(t|r)$$

Operation 2: Eliminate

- Second basic operation: **marginalization**
- Take a factor and sum out a variable
  - Shrinks a factor to a smaller one
  - A projection operation

- Example:

$P(R, T)$

+r	+t	0.08
+r	-t	0.02
-r	+t	0.09
-r	-t	0.81

sum R →

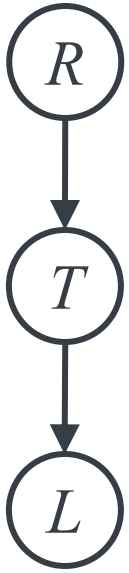
$P(T)$

+t	0.17
-t	0.83

Variable Elimination

- Thus Far: Multiple Join, Multiple Eliminate
  - Inference by Enumeration
- Marginalizing Early
  - Variable Elimination

Traffic Domain



$$P(L) = ?$$

- Inference by Enumeration

$$= \sum_t \sum_r P(L|t)P(r)P(t|r)$$

Join on r

Join on t

Eliminate r

Eliminate t

- Variable Elimination

$$= \sum_t P(L|t) \sum_r P(r)P(t|r)$$

Join on r

Eliminate r

Join on t

Eliminate t

General Variable Elimination

- Query: $P(Q|E_1 = e_1, \ldots E_k = e_k)$

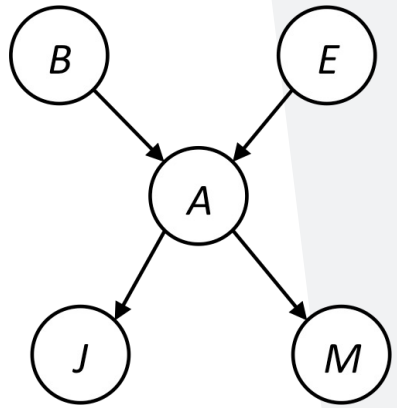
- Start with initial factors:
  - Local CPTs (but instantiated by evidence)

- While there are still hidden variables (not Q or evidence):
  - Pick a hidden variable H
  - Join all factors mentioning H
  - Eliminate (sum out) H

- Join all remaining factors and normalize

Example

$$P(B|j,m) \propto P(B,j,m)$$



$P(B)$	$P(E)$	$P(A\|B,E)$	$P(j\|A)$	$P(m\|A)$

$$\begin{aligned}
P(B|j,m) &\propto P(B,j,m) \\
&= \sum_{e,a} P(B,j,m,e,a) && \text{marginal can be obtained from joint by summing out} \\
&= \sum_{e,a} P(B)P(e)P(a|B,e)P(j|a)P(m|a) && \text{use Bayes' net joint distribution expression} \\
&= \sum_{e} P(B)P(e)\sum_{a} P(a|B,e)P(j|a)P(m|a) && \text{use } x*(y+z) = xy + xz \\
&= \sum_{e} P(B)P(e)f_1(B,e,j,m) && \text{joining on a, and then summing out gives } f_1 \\
&= P(B)\sum_{e} P(e)f_1(B,e,j,m) && \text{use } x*(y+z) = xy + xz \\
&= P(B)f_2(B,j,m) && \text{joining on e, and then summing out gives } f_2
\end{aligned}$$

All we are doing is exploiting uwy + uwz + uxy + uxz + vwy + vwz + vxy +vxz = (u+v)(w+x)(y+z) to improve computational efficiency!

STEVENS
INSTITUTE OF TECHNOLOGY
1870

Bayesian Network

Chapter 13, 14

Another Variable Elimination Example

Query: $P(X_3|Y_1 = y_1, Y_2 = y_2, Y_3 = y_3)$

Start by inserting evidence, which gives the following initial factors:

$$p(Z)p(X_1|Z)p(X_2|Z)p(X_3|Z)p(y_1|X_1)p(y_2|X_2)p(y_3|X_3)$$

Eliminate X_1, this introduces the factor $f_1(Z, y_1) = \sum_{x_1} p(x_1|Z)p(y_1|x_1)$, and we are left with:

$$p(Z)f_1(Z, y_1)p(X_2|Z)p(X_3|Z)p(y_2|X_2)p(y_3|X_3)$$

Eliminate X_2, this introduces the factor $f_2(Z, y_2) = \sum_{x_2} p(x_2|Z)p(y_2|x_2)$, and we are left with:

$$p(Z)f_1(Z, y_1)f_2(Z, y_2)p(X_3|Z)p(y_3|X_3)$$

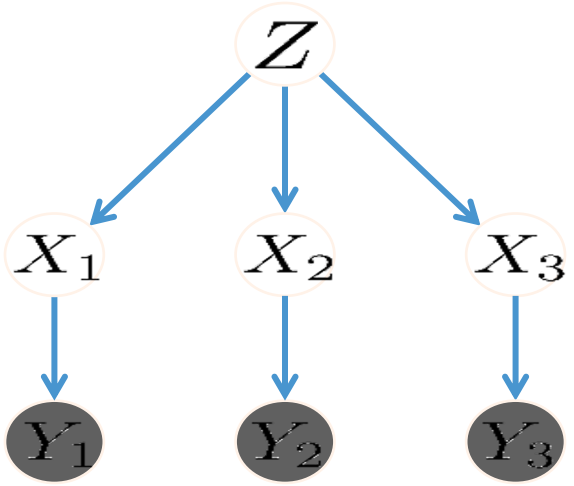
Eliminate Z, this introduces the factor $f_3(y_1, y_2, X_3) = \sum_z p(z)f_1(z, y_1)f_2(z, y_2)p(X_3|z)$, and we are left:

$$p(y_3|X_3), f_3(y_1, y_2, X_3)$$

No hidden variables left. Join the remaining factors to get:

$$f_4(y_1, y_2, y_3, X_3) = P(y_3|X_3)f_3(y_1, y_2, X_3).$$

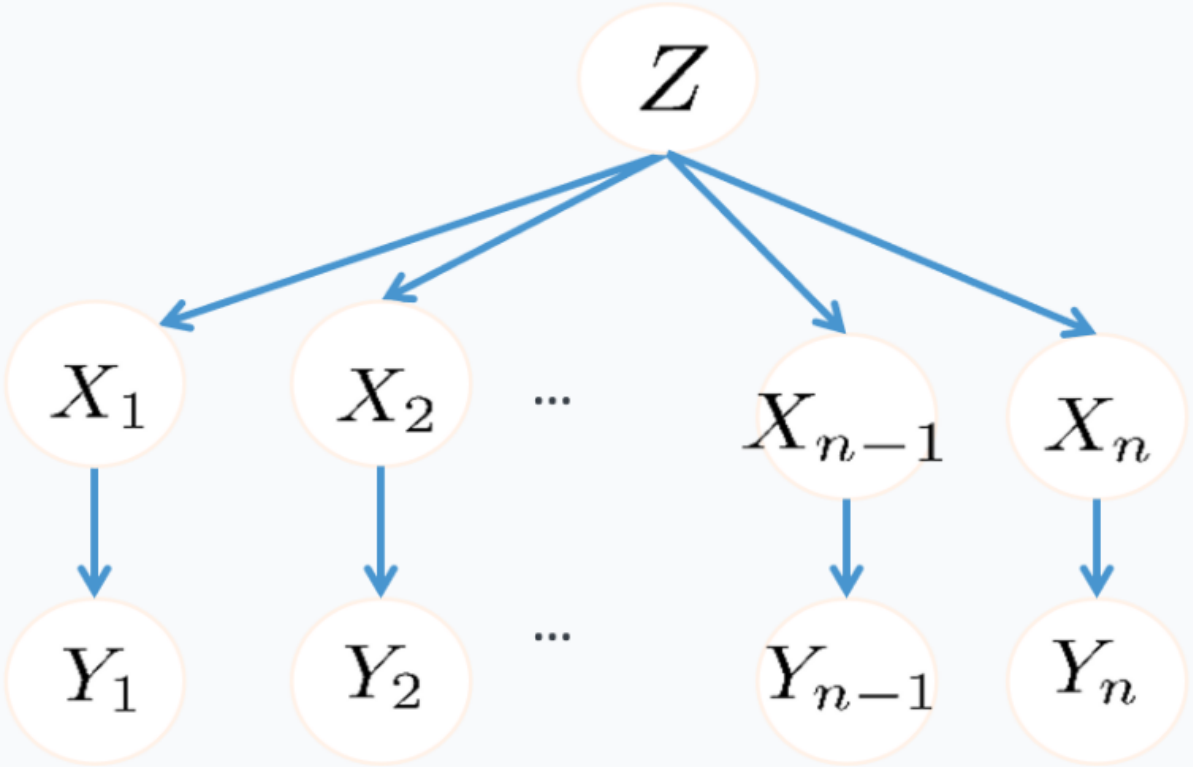
Normalizing over X_3 gives $P(X_3|y_1, y_2, y_3)$.



Computational complexity critically depends on the largest factor being generated in this process.

- Size of factor = number of entries in table.
- In example above (assuming binary) all factors generated are of size 2 --- as they all only have one.

For the query P(Xn|y1,…,yn), which ordering is more efficient?



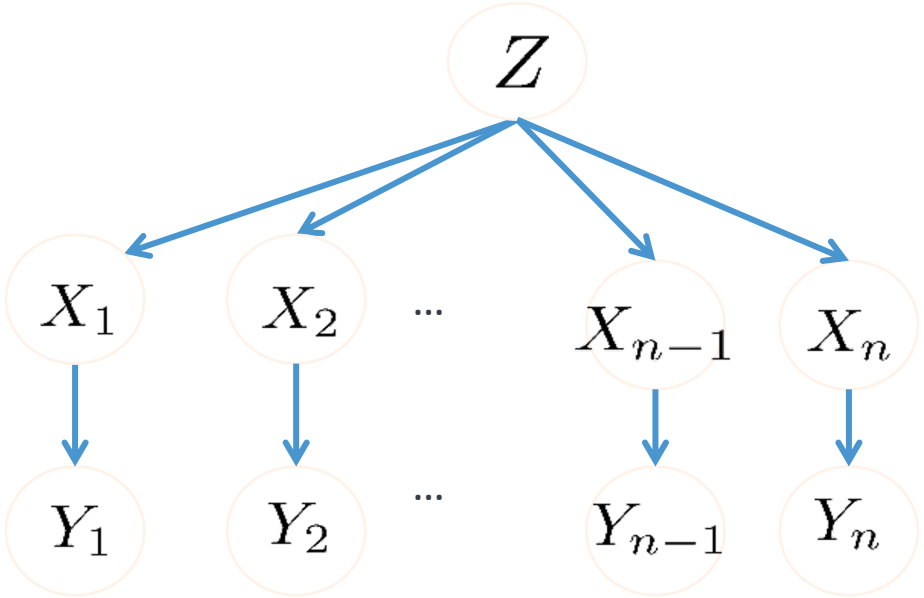
Z, X1, X2,…,Xn-1

X1, X2,…,Xn-1,Z

The ordering does not affect the efficiency.

Variable Elimination Ordering

- For the query $P(X_n|y_1,...,y_n)$ work through the following two different orderings as done in previous slide: Z, X_1, …, X_{n-1} and X_1, …, X_{n-1}, Z. What is the size of the maximum factor generated for each of the orderings?



- Answer: 2^{n+1} versus 2^2 (assuming binary)
- In general: the ordering can greatly affect efficiency.

VE: Computational and Space Complexity

- The computational and space complexity of variable elimination is determined by the largest factor.
- The elimination ordering can greatly affect the size of the largest factor.
  - E.g., previous slide's example 2^{n+1} vs. 2^2
- Does there always exist an ordering that only results in small factors?
  - **No!**

Worst Case Complexity?

- CSP:

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (x_2 \vee \neg x_2 \vee x_4) \wedge (\neg x_3 \vee \neg x_4 \vee \neg x_5) \wedge (x_2 \vee x_5 \vee x_7) \wedge (x_4 \vee x_5 \vee x_6) \wedge (\neg x_5 \vee x_6 \vee \neg x_7) \wedge (\neg x_5 \vee \neg x_6 \vee x_7)$$

$$P(X_i = 0) = P(X_i = 1) = 0.5$$

$$Y_1 = X_1 \vee X_2 \vee \neg X_3$$

$$\cdots$$

$$Y_8 = \neg X_5 \vee X_6 \vee X_7$$

$$Y_{1,2} = Y_1 \wedge Y_2$$

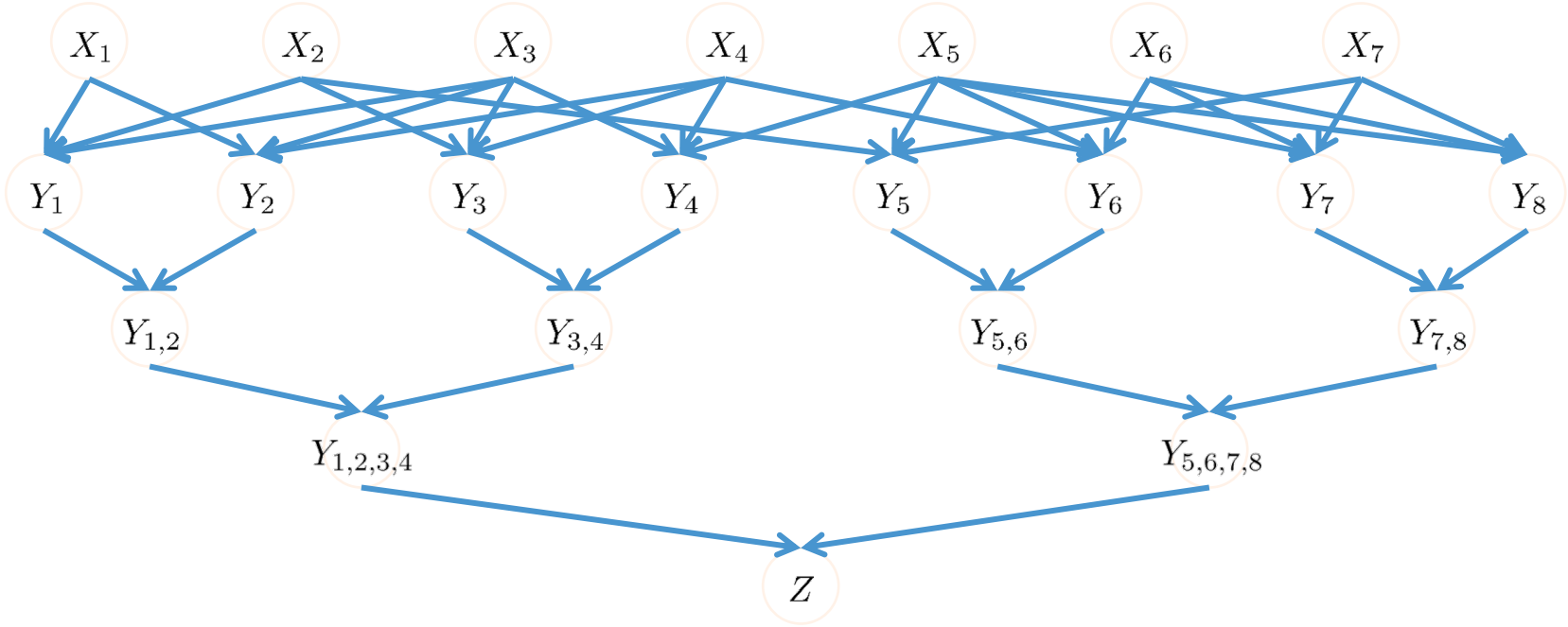
$$\cdots$$

$$Y_{7,8} = Y_7 \wedge Y_8$$

$$Y_{1,2,3,4} = Y_{1,2} \wedge Y_{3,4}$$

$$Y_{5,6,7,8} = Y_{5,6} \wedge Y_{7,8}$$

$$Z = Y_{1,2,3,4} \wedge Y_{5,6,7,8}$$

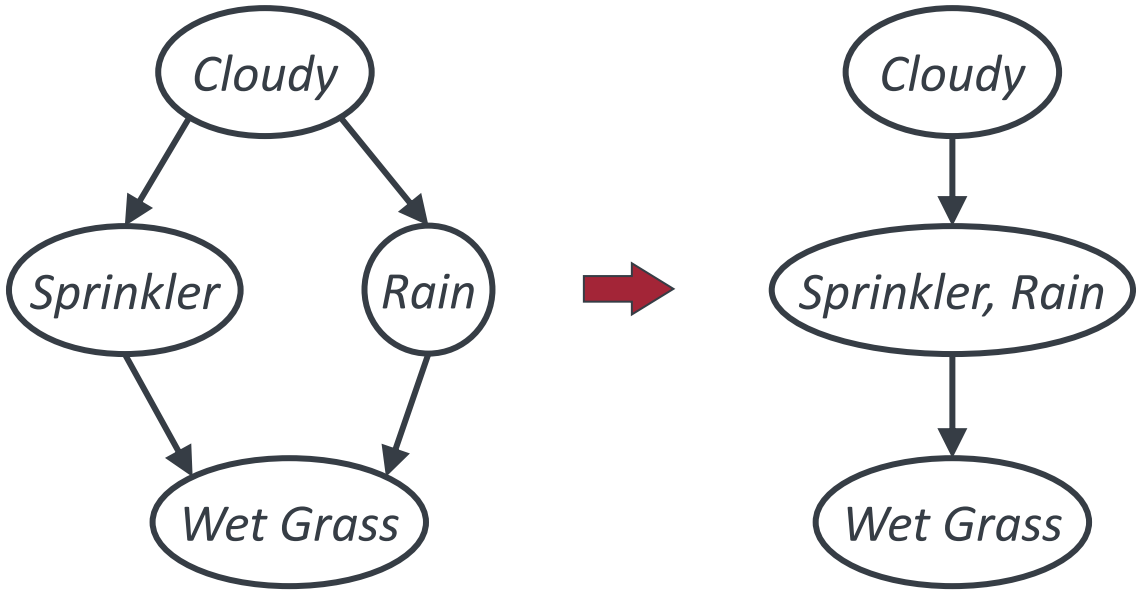


- If we can answer P(z) equal to zero or not, we answered whether the **3-SAT problem** has a solution.
- Hence inference in Bayes' nets is NP-hard. No known efficient probabilistic inference in general.

Polytrees

- A polytree is a directed graph with no undirected cycles.
- For poly-trees you can always find an ordering that is efficient.
  - The time and space complexity of exact inference in polytrees is linear in the size of the network.
- Cut-set conditioning for Bayes' net inference
  - Choose set of variables such that if removed only a polytree remains
  - Exercise: Think about how the specifics would work out!

Polytrees



- The graph can be converted to a polytree by merging the variables.
  - The size of the CPTs increases.
- It is always possible to build a join tree but may require exponentially many combinations. of values.

Bayes' Nets

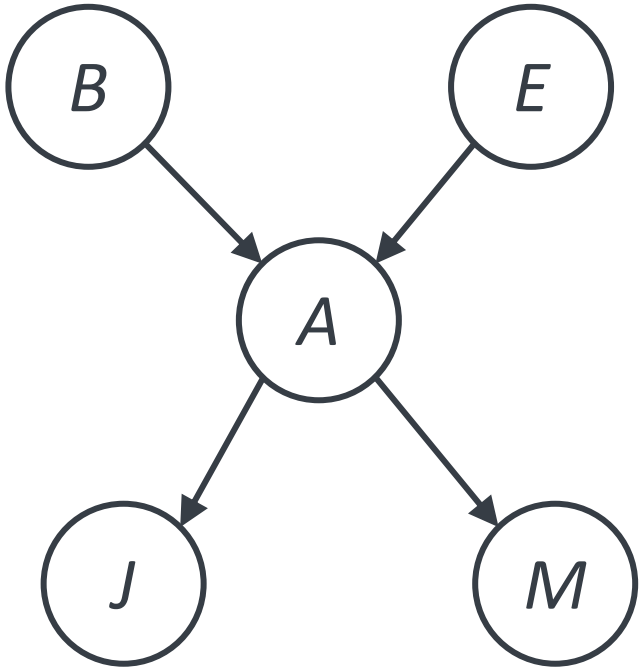
- Representation
- Conditional Independences
- Probabilistic Inference
  - Enumeration (exact, exponential complexity)
  - Variable elimination (exact, worst-case exponential complexity, often better)
  - Probabilistic inference is NP-complete
  - Sampling (approximate)
- Learning Bayes' Nets from Data

Approximate Inference: Sampling

Inference

- What is P(B|+j, +m)?

$$P(Q|e_1 \dots e_k)$$



Sampling

- Sampling is a lot like repeated simulation.
  - Predicting the weather, basketball games, …
- Basic idea
  - Draw N samples from a sampling distribution S.
  - Compute an approximate posterior probability.
  - Show this converges to the true probability P.
- Why sample?
  - Learning: get samples from a distribution you don’t know
  - Inference: getting a sample is faster than computing the right answer (*e.g.*, with variable elimination)

Sampling

- Sampling from given distribution
  - Step 1: Get sample *u* from uniform distribution over [0, 1)
    - E.g., random() in python
  - Step 2: Convert this sample *u* into an outcome for the given distribution by having each outcome associated with a sub-interval of [0,1) with sub-interval size equal to probability of the outcome

- Example

C	P(C)
red	0.6
green	0.1
blue	0.3

$$0 \leq u < 0.6, \rightarrow C = red$$
$$0.6 \leq u < 0.7, \rightarrow C = green$$
$$0.7 \leq u < 1, \rightarrow C = blue$$

- If random() returns $u = 0.83$, then our sample is C = blue
- E.g., after sampling 8 times:
  - 5 red, 1 green, 2 blue

Sampling in Bayes' Nets

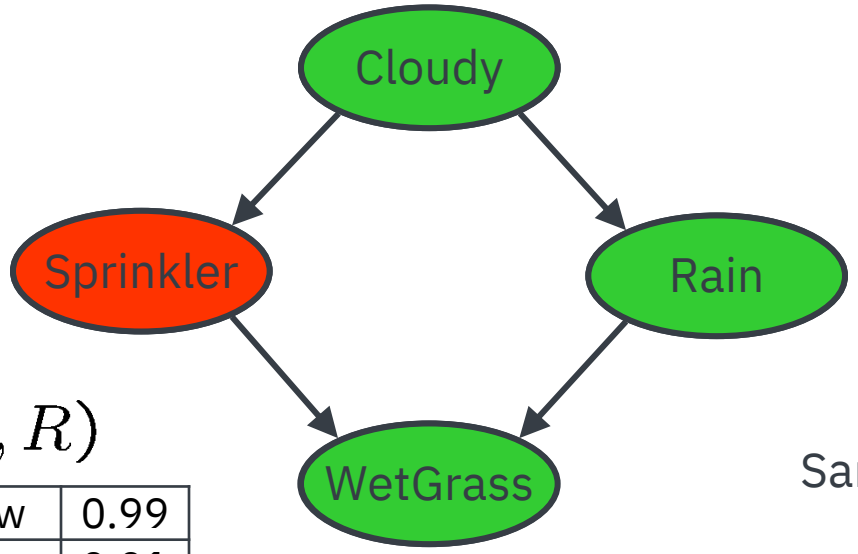
- Prior Sampling
- Rejection Sampling
- Likelihood Weighting
- Gibbs Sampling
- …

Prior Sampling

Prior Sampling

$P(C)$

+c	0.5
-c	0.5



$P(S|C)$

+c	+s	0.1
	-s	0.9
-c	+s	0.5
	-s	0.5

$P(R|C)$

+c	+r	0.8
	-r	0.2
-c	+r	0.2
	-r	0.8

$P(W|S,R)$

+s	+r	+w	0.99
		-w	0.01
	-r	+w	0.90
		-w	0.10
-s	+r	+w	0.90
		-w	0.10
	-r	+w	0.01
		-w	0.99

Samples:

+c, -s, +r, +w

-c, +s, -r, +w

…

Compute $P(Q|e_1 \ldots e_k)$ from samples.

Prior Sampling

- For i=1, 2, …, n
  - Sample x_i from $P(X_i \mid \text{Parents}(X_i))$
- Return $(x_1, x_2, \ldots, x_n)$

Prior Sampling

- This process generates samples with probability:

$$S_{PS}(x_1 \ldots x_n) = \prod_{i=1}^{n} P(x_i | \text{Parents}(X_i)) = P(x_1 \ldots x_n)$$

...i.e., the BN's joint probability

- Let the number of samples of an event be $N_{PS}(x_1 \ldots x_n)$

- Then

$$\begin{aligned} \lim_{N \to \infty} \hat{P}(x_1, \ldots, x_n) &= \lim_{N \to \infty} N_{PS}(x_1, \ldots, x_n)/N \\ &= S_{PS}(x_1, \ldots, x_n) \\ &= P(x_1 \ldots x_n) \end{aligned}$$

- I.e., the sampling procedure is **consistent.**

Example

- We will get a bunch of samples from the BN:

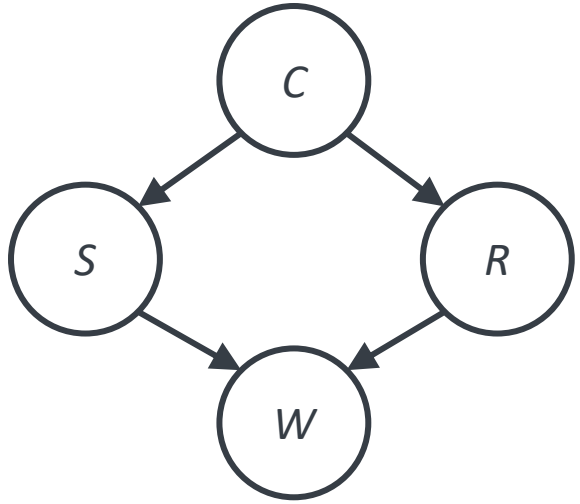
  +c, -s, +r, +w

  +c, +s, +r, +w

  -c, +s, +r, -w

  +c, -s, +r, +w

  -c, -s, -r, +w



- If we want to know P(W)
  - We have counts <+w:4, -w:1>
  - Normalize to get P(W) = <+w:0.8, -w:0.2>
  - This will get closer to the true distribution with more samples.
  - Can estimate anything else, too.
  - What about P(C| +w)? P(C| +r, +w)? P(C| -r, -w)?
  - Fast: can use fewer samples if less time (what's the drawback?)

Can we make it more efficient?

What does the value N(+a, -b, +c)/N approximate?

- $P(+a, -b, +c)$
- $P(+c \mid +a, -b)$
- $P(+c \mid -b)$
- $P(+c)$

How many {-a, +b, -c} samples out of N=1000 should we expect?

P(A)

+a	0.5
-a	0.5

P(B|A)

+a	+b	0.1
	-b	0.9
-a	+b	0.5
	-b	0.5

P(C|B)

+b	+c	0.8
	-c	0.2
-b	+c	1.0
	-c	0.0

1

40

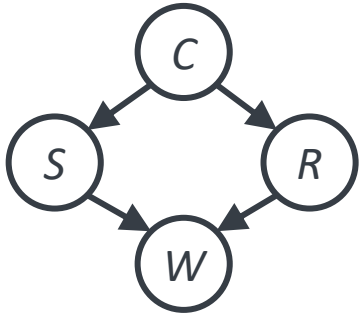
125

200

Rejection Sampling

Rejection Sampling

- Let's say we want P(C)
  - No point keeping all samples around
  - Just tally counts of C as we go



- Let's say we want P(C| +s)
  - Same thing: tally C outcomes, but ignore(reject) samples which don't have S=+s.
  - This is called rejection sampling.
  - It is also consistent for conditional probabilities (i.e., correct in the limit).

+c, -s, +r, +w
+c, +s, +r, +w
-c, +s, +r, -w
+c, -s, +r, +w
-c, -s, -r, +w

Rejection Sampling

- IN: evidence instantiation
- For i=1, 2, …, n
  - Sample x_i from $P(X_i \mid Parents(X_i))$
  - If x_i not consistent with evidence
    - Reject: Return, and no sample is generated in this cycle
- Return $(x_1, x_2, \ldots, x_n)$

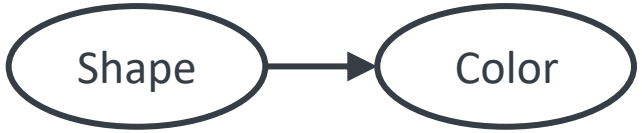
What queries can we answer with rejection samples (evidence: +c)?

P(+a, -b, +c)
P(+a, -b | +c)
Both
Neither

Likelihood Weighting

Likelihood Weighting

- Problem with rejection sampling:
  - If evidence is unlikely, rejects lots of samples
  - Evidence not exploited as you sample
  - Consider P(Shape|blue)



~~pyramid, green~~
~~pyramid, red~~
sphere, blue
~~cube, red~~
~~sphere,~~
~~green~~

- Idea: fix evidence variables and sample the rest
  - Problem: sample distribution not consistent!
  - Solution: weight by probability of evidence given parents

pyramid, blue
pyramid, blue
sphere, blue
cube, blue
sphere, blue

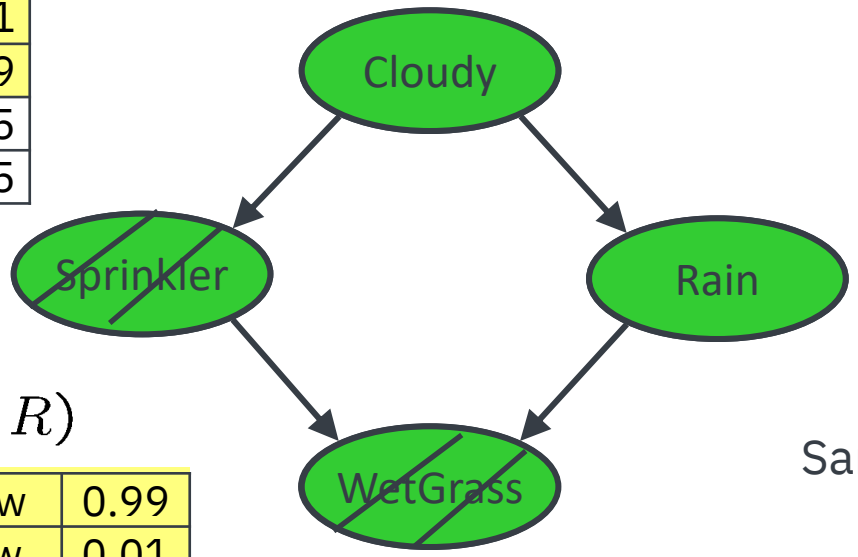
Likelihood Weighting

$P(C)$

+c	0.5
-c	0.5

$P(S|C)$

+c	+s	0.1
	-s	0.9
-c	+s	0.5
	-s	0.5



$P(R|C)$

+c	+r	0.8
	-r	0.2
-c	+r	0.2
	-r	0.8

$P(W|S,R)$

+s	+r	+w	0.99
		-w	0.01
	-r	+w	0.90
		-w	0.10
-s	+r	+w	0.90
		-w	0.10
	-r	+w	0.01
		-w	0.99

Samples:

+c, +s, +r, +w

…

$$w = 1.0 \times 0.1 \times 0.99$$

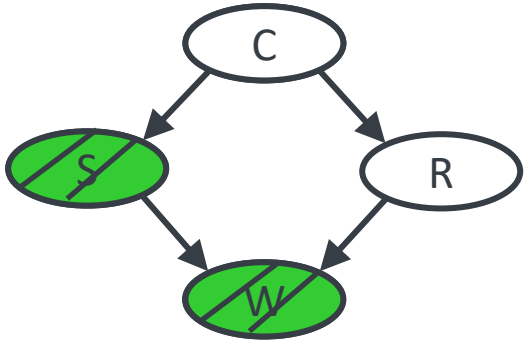
Likelihood Weighting

- IN: evidence instantiation
- w = 1.0
- for i=1, 2, …, n
  - if X_i is an evidence variable
    - X_i = observation x_i for X_i
    - Set w = w * $P(x_i \mid Parents(X_i))$
  - else
    - Sample x_i from $P(X_i \mid Parents(X_i))$
- return $(x_1, x_2, \ldots, x_n)$, w

Likelihood Weighting

- Sampling distribution if z sampled and e fixed evidence

$$S_{WS}(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^{l} P(z_i | \text{Parents}(Z_i))$$



- Now, samples have weights

$$w(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^{m} P(e_i | \text{Parents}(E_i))$$

- Together, weighted sampling distribution is consistent

$$S_{\text{WS}}(z, e) \cdot w(z, e) = \prod_{i=1}^{l} P(z_i | \text{Parents}(z_i)) \prod_{i=1}^{m} P(e_i | \text{Parents}(e_i))$$

$$= P(\mathbf{z}, \mathbf{e})$$

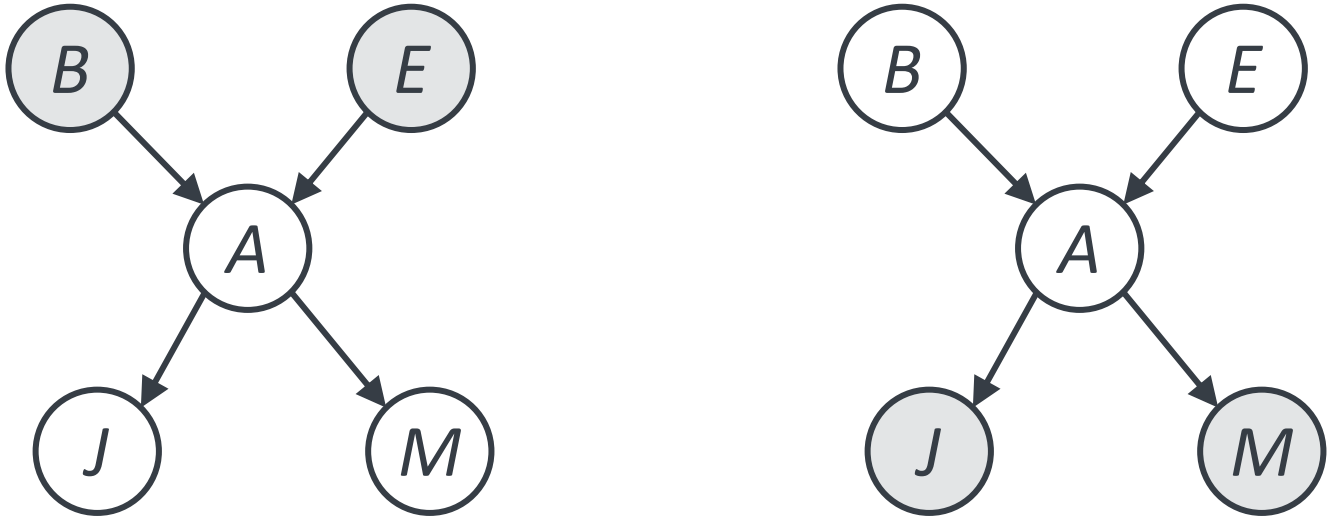
Likelihood Weighting

- Likelihood weighting is good
  - We have taken evidence into account as we generate the sample
  - E.g., here, W’ s value will get picked based on the evidence values of S, R
  - More of our samples will reflect the state of the world suggested by the evidence
- Likelihood weighting doesn’t solve all our problems
  - Evidence influences the choice of downstream variables, but not upstream ones (C isn’t more likely to get a value matching the evidence)
- We would like to consider evidence when we sample every variable

  → Gibbs sampling

Limitations of Likelihood Weighting

- Only conditions on upstream evidence, and hence weights obtained in likelihood weighting can sometimes be very small.
- Sum of weights over all samples is indicative of how many “effective” samples were obtained, so want high weight.



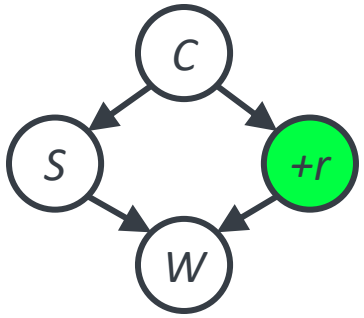
Gibbs Sampling

Gibbs Sampling

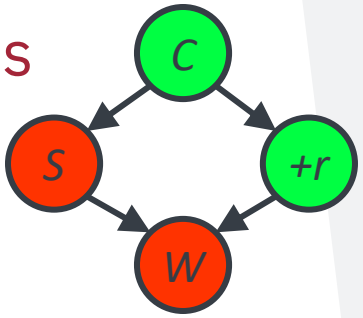
- Procedure
  1. Keep track of a full instantiation $x_1, x_2, \ldots, x_n$.
  2. Start with an arbitrary instantiation consistent with the evidence.
  3. Sample one variable at a time, conditioned on all the rest, but keep evidence fixed.
  4. Keep repeating this for a long time.
- Property
  - In the limit of repeating this infinitely many times, the resulting sample is coming from the correct distribution.
- Rationale
  - Both upstream and downstream variables condition on evidence.

Gibbs Sampling Example: P(S | +r)

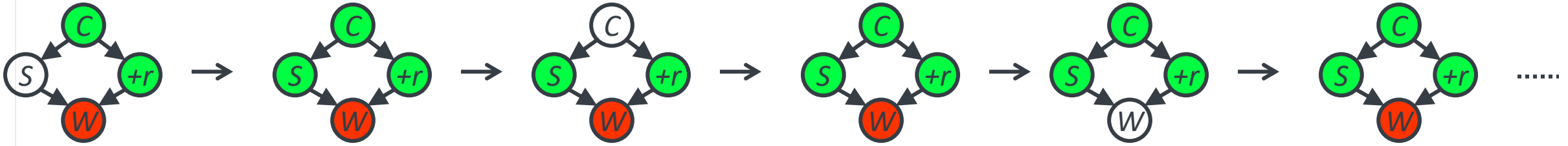
- Step 1: Fix evidence
  - R = +r



- Step 2: Initialize other variables
  - Randomly



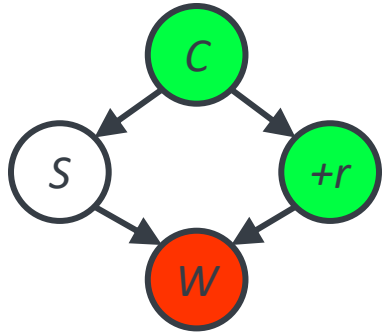
- Steps 3: Repeat
  - Choose a non-evidence variable X
  - Resample X from P(X | all other variables)



Sample from $P(S|+c,-w,+r)$ Sample from $P(C|+s,-w,+r)$ Sample from $P(W|+s,+c,+r)$

Efficient Resampling of One Variable

- Sample from P(S | +c, +r, -w)



$$
\begin{aligned}
P(S|+c,+r,-w) &= \frac{P(S,+c,+r,-w)}{P(+c,+r,-w)} \\
&= \frac{P(S,+c,+r,-w)}{\sum_s P(s,+c,+r,-w)} \\
&= \frac{P(+c)P(S|+c)P(+r|+c)P(-w|S,+r)}{\sum_s P(+c)P(s|+c)P(+r|+c)P(-w|s,+r)} \\
&= \frac{P(+c)P(S|+c)P(+r|+c)P(-w|S,+r)}{P(+c)P(+r|+c)\sum_s P(s|+c)P(-w|s,+r)} \\
&= \frac{P(S|+c)P(-w|S,+r)}{\sum_s P(s|+c)P(-w|s,+r)}
\end{aligned}
$$

- Many things cancel out – only CPTs with S remain!
- More generally: only CPTs that have resampled variable need to be considered, and joined together

Further Reading on Gibbs Sampling

- Gibbs sampling produces sample from the query distribution P(Q | e) in limit of re-sampling infinitely often.
- Gibbs sampling is a special case of more general methods called **Markov chain Monte Carlo (MCMC)** methods.
  - Metropolis-Hastings is one of the more famous MCMC methods (in fact, Gibbs sampling is a special case of Metropolis-Hastings).
- You may read about Monte Carlo methods – they're just sampling.

Pros and Cons of Gibbs Sampling

- Advantages
  - No samples are discarded.
  - No problem with samples with low weight.
  - Can be implemented very efficiently.
- Disadvantages
  - Depending on the target distribution, there may be very strong correlations between consecutive samples.
  - To get less dependence, Gibbs sampling is often run for a long time, and the samples are thinned by keeping only every 10^{th} or 100^{th} sample.

Bayes' Net Sampling Summary

- Prior Sampling: **P**
- Rejection Sampling: **P(Q | e)**
- Likelihood Weighting: **P(Q | e)**
- Gibbs Sampling: **P(Q | e)**

STEVENS
INSTITUTE OF TECHNOLOGY
1870

15 min. break

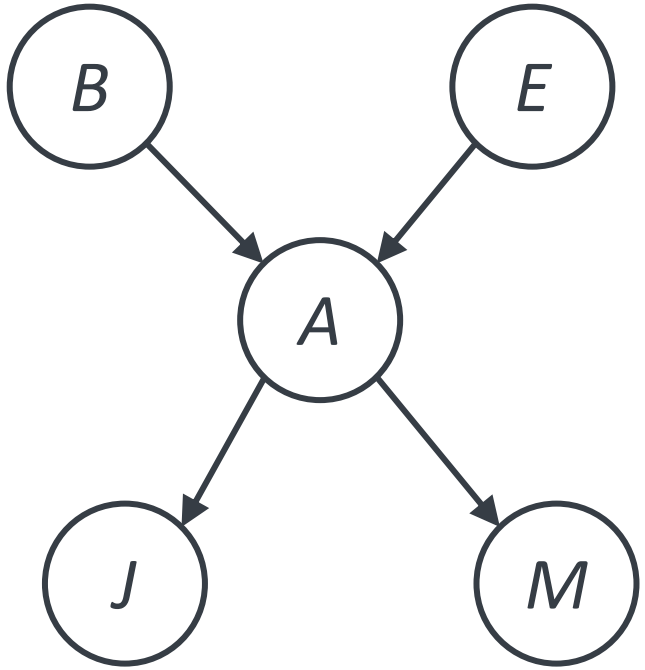
Machine Learning

- Up until now: how use a model to make optimal decisions
- Machine learning: how to acquire a model from data and experience
  - Learning parameters (e.g., probabilities)
  - Learning structure (e.g., BN graphs)
  - Learning hidden concepts (e.g., clustering)

- Today: model-based classification with Naive Bayes

Burglars and Earthquakes

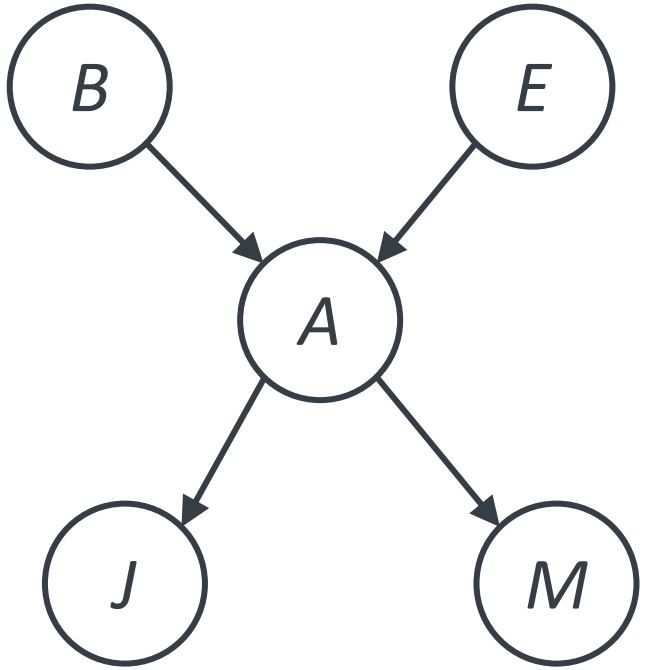
Earthquake	Burglars	Alarm	#
0	0	0	1000
0	0	1	10
0	1	0	20
0	1	1	100
1	0	0	200
1	0	1	50
1	1	0	0
1	1	1	5



	P(+a|E, B)
+e, +b	
+e, -b	
-e, +b	
-e, -b	

Burglars and Earthquakes

Earthquake	Burglars	Alarm	#
0	0	0	1000
0	0	1	10
0	1	0	20
0	1	1	100
1	0	0	200
1	0	1	50
1	1	0	0
1	1	1	5



	P(+a|E, B)
+e, +b	1
+e, -b	0.2
-e, +b	0.83
-e, -b	0.01

Classification

Example: Spam Filter

- Input: an email
- Output: spam/ham

- Setup:
  - Get a large collection of example emails, each labeled “spam” or “ham”
  - Note: someone has to hand label all this data!
  - Want to learn to predict labels of new, future emails

- Features: The attributes used to make the ham / spam decision
  - Words: FREE!
  - Text Patterns: $dd, CAPS
  - Non-text: SenderInContacts
  - …

Dear Sir.

First, I must solicit your confidence in this transaction, this is by virture of its nature as being utterly confidencial and top secret. …

TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.

99 MILLION EMAIL ADDRESSES FOR ONLY $99

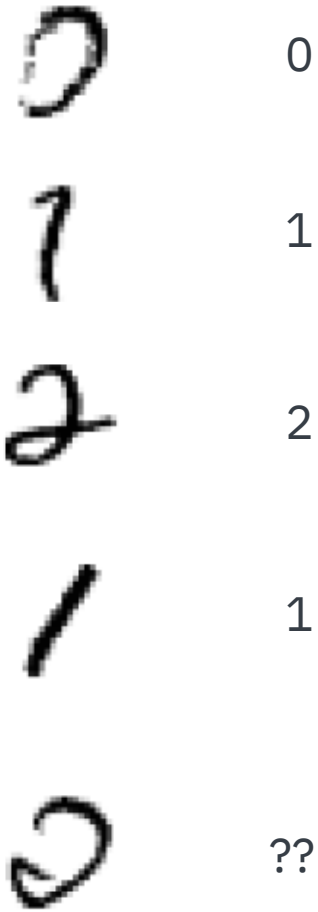
Ok, Iknow this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.

Example: Digit Recognition

- Input: images / pixel grids
- Output: a digit 0-9

- Setup:
  - Get a large collection of example images, each labeled with a digit
  - Note: someone has to hand label all this data!
  - Want to learn to predict labels of new, future digit images

- Features: The attributes used to make the digit decision
  - Pixels: (6,8)=ON
  - Shape Patterns: NumComponents, AspectRatio, NumLoops
  - …



Other Classification Tasks

- Classification: given inputs x, predict labels (classes) y
- Examples:
  - Spam detection (input: document, classes: spam / ham)
  - OCR (input: images, classes: characters)
  - Medical diagnosis (input: symptoms, classes: diseases)
  - Automatic essay grading (input: document, classes: grades)
  - Fraud detection (input: account activity, classes: fraud / no fraud)
  - Customer service email routing
  - ... many more

- Classification is an important commercial technology!

Model-Based Classification

Model-Based Classification

- Model-based approach
  - Build a model (e.g., Bayes' net) where both the label and features are random variables
  - Instantiate any observed features
  - Query for the distribution of the label conditioned on the features

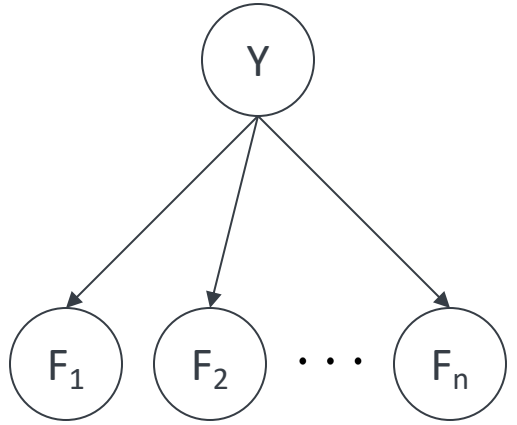
- Challenges
  - What structure should the BN have?
  - How should we learn its parameters?

Naïve Bayes for Digits

- Naïve Bayes: Assume all features are independent effects of the label
- Simple digit recognition version:
  - One feature (variable) F_{ij} for each grid position <i,j>
  - Feature values are on / off, based on whether intensity is more or less than 0.5 in underlying image
  - Each input maps to a feature vector, e.g.

$$\rightarrow \langle F_{0,0}=0 \;\; F_{0,1}=0 \;\; F_{0,2}=1 \;\; F_{0,3}=1 \;\; F_{0,4}=0 \;\; \ldots F_{15,15}=0 \rangle$$

  - Here: lots of features, each is binary valued



- Naïve Bayes model: $P(Y|F_{0,0} \ldots F_{15,15}) \propto P(Y) \prod_{i,j} P(F_{i,j}|Y)$
- What do we need to learn?

General Naïve Bayes

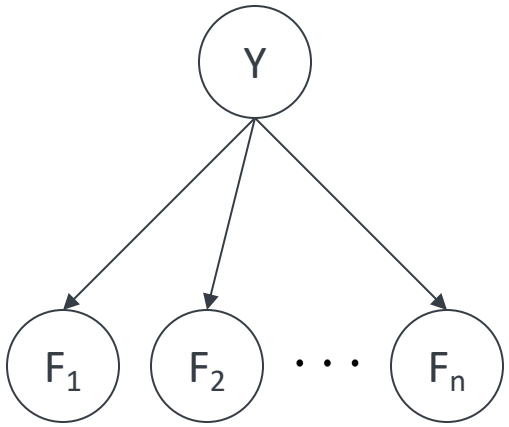
- A general **Naive Bayes** model:

|Y| parameters

$$P(\mathrm{Y}, \mathrm{F}_1 \ldots \mathrm{F}_n) = P(\mathrm{Y}) \prod_i P(\mathrm{F}_i|\mathrm{Y})$$

|Y| x |F|^n values

n x |F| x |Y| parameters



- We only have to specify how each feature depends on the class
- Total number of parameters is **linear** in n
- Model is very simplistic, but often works anyway

Inference for Naïve Bayes

- Goal: compute posterior distribution over label variable Y
  - Step 1: get joint probability of label and evidence for each label

$$P(Y, f_1 \dots f_n) = \begin{bmatrix} P(y_1, f_1 \dots f_n) \\ P(y_2, f_1 \dots f_n) \\ \vdots \\ P(y_k, f_1 \dots f_n) \end{bmatrix} \Rightarrow \begin{bmatrix} P(y_1) \prod_i P(f_i|y_1) \\ P(y_2) \prod_i P(f_i|y_2) \\ \vdots \\ P(y_k) \prod_i P(f_i|y_k) \end{bmatrix}$$

$$\Rightarrow \frac{\begin{bmatrix} P(y_1) \prod_i P(f_i|y_1) \\ P(y_2) \prod_i P(f_i|y_2) \\ \vdots \\ P(y_k) \prod_i P(f_i|y_k) \end{bmatrix}}{P(f_1 \dots f_n)} \quad (+) \Rightarrow P(Y|f_1 \dots f_n)$$

  - Step 2: sum to get probability of evidence
  - Step 3: normalize by dividing Step 1 by Step 2

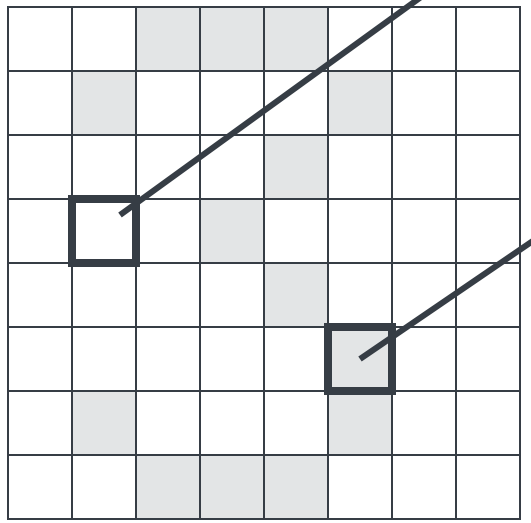
General Naïve Bayes

- What do we need to use Naïve Bayes?
  - Inference method (we just saw this part)
    - Start with a bunch of probabilities: P(Y) and the $P(F_i|Y)$ tables
    - Use standard inference to compute $P(Y|F_1...F_n)$
    - Nothing new here
  - Estimates of local conditional probability tables
    - P(Y), the prior over labels
    - $P(F_i|Y)$ for each feature (evidence variable)
    - These probabilities are collectively called the **parameters** of the model and denoted by θ
    - Up until now, we assumed these appeared by magic, but…
    - …they typically come from training data counts: we'll look at this soon

Example: Conditional Probabilities

$P(Y)$

1	0.1
2	0.1
3	0.1
4	0.1
5	0.1
6	0.1
7	0.1
8	0.1
9	0.1
0	0.1



$P(F_{3,1} = on|Y)$

1	0.01
2	0.05
3	0.05
4	0.30
5	0.80
6	0.90
7	0.05
8	0.60
9	0.50
0	0.80

$P(F_{5,5} = on|Y)$

1	0.05
2	0.01
3	0.90
4	0.80
5	0.90
6	0.90
7	0.25
8	0.85
9	0.60
0	0.80

Naïve Bayes for Text

- Bag-of-words Naïve Bayes:
  - Features: W_i is the word at positon i
  - As before: predict label conditioned on feature variables (spam vs. ham)
  - As before: assume features are conditionally independent given label
  - New: each W_i is identically distributed
- Generative model: $P(Y, W_1 \dots W_n) = P(Y) \prod_i P(W_i|Y)$

Word at position i, not i^{th} word in the dictionary!

- "Tied" distributions and bag-of-words
  - Usually, each variable gets its own conditional probability distribution P(F|Y)
  - In a bag-of-words model
    - Each position is identically distributed
    - All positions share the same conditional probs P(W|Y)
  - Called "bag-of-words" because model is insensitive to word order or reordering

Example: Spam Filtering

- Model: $P(Y, W_1 \dots W_n) = P(Y) \prod_i P(W_i|Y)$
- What are the parameters?

$P(Y)$

ham	0.66
spam	0.33

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

the	0.0156
to	0.0153
and	0.0115
of	0.0095
you	0.0093
a	0.0086
with	0.0080
from	0.0075
...	

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

the	0.0210
to	0.0133
of	0.0119
2002	0.0110
with	0.0108
from	0.0107
and	0.0105
a	0.0100
...	

- Where do these tables come from?

Spam Example

Word	P(w|spam)	P(w|ham)	Tot Spam	Tot Ham
(prior)	0.33333	0.66666	-1.1	-0.4

STEVENS
INSTITUTE OF TECHNOLOGY
1870

Learning Probabilistic Models

Chapter 20

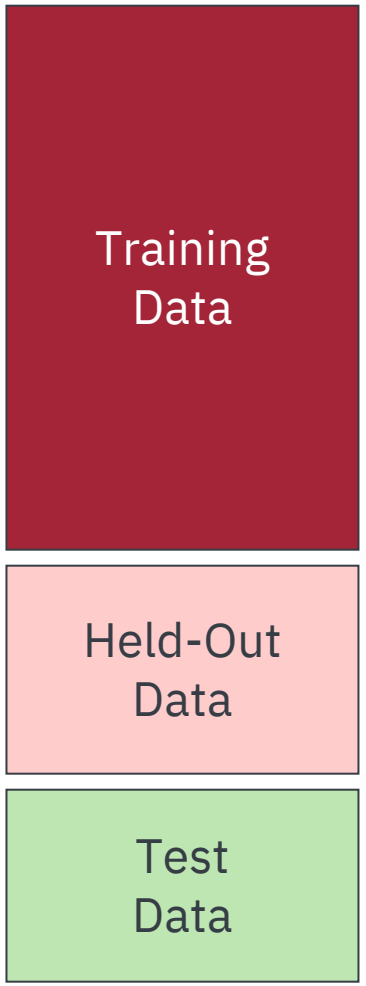
Training and Testing

Empirical Risk Minimization

- Empirical risk minimization
  - Basic principle of machine learning
  - We want to model (classifier, etc) that does best on the true test distribution.
  - Don’t know the true distribution so pick the best model on our actual training set.
  - Finding **the best model** on the training set is phrased as an optimization problem.
- Main worry: overfitting to the training set
  - Better with more training data (less sampling variance, training more like test)
  - Better if we limit the complexity of our hypotheses (regularization and/or small hypothesis spaces)

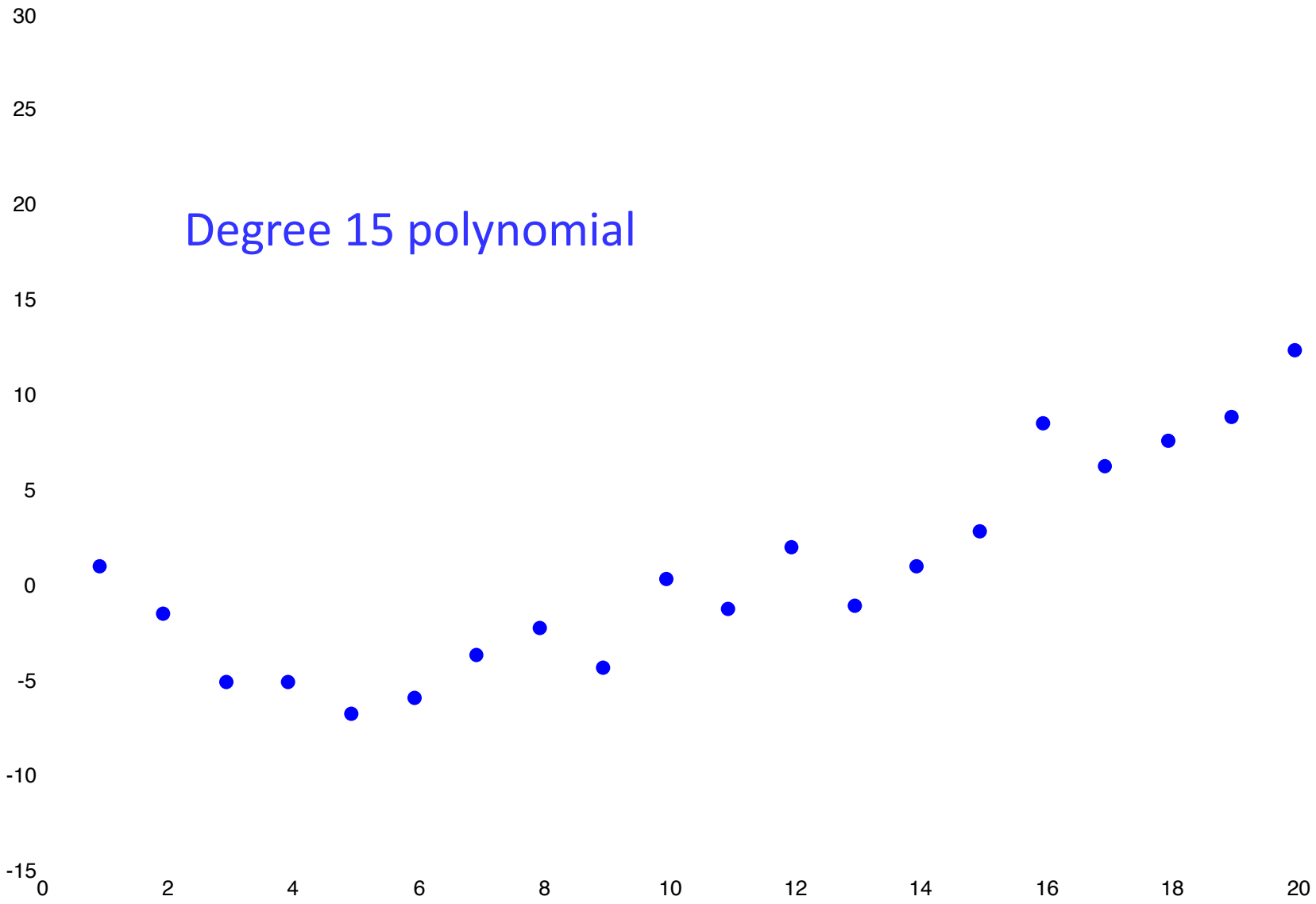
Important Concepts

- Data: labeled instances, *e.g.,* emails marked spam/ham
  - Training set
  - Held out set
  - Test set
- Features: attribute-value pairs which characterize each x
- Experimentation cycle
  - Learn parameters (*e.g.,* model probabilities) on training set
  - (Tune hyperparameters on held-out set)
  - Compute accuracy of test setEvaluation
  - Accuracy: fraction of instances predicted correctly
- Overfitting and generalization
  - Want a classifier which does well on *test* data
  - Overfitting: fitting the training data very closely, but not generalizing well
  - We'll investigate overfitting and generalization formally.



Generalization and Overfitting

Overfitting



Example: Overfitting

$P(\text{features}, C = 2)$

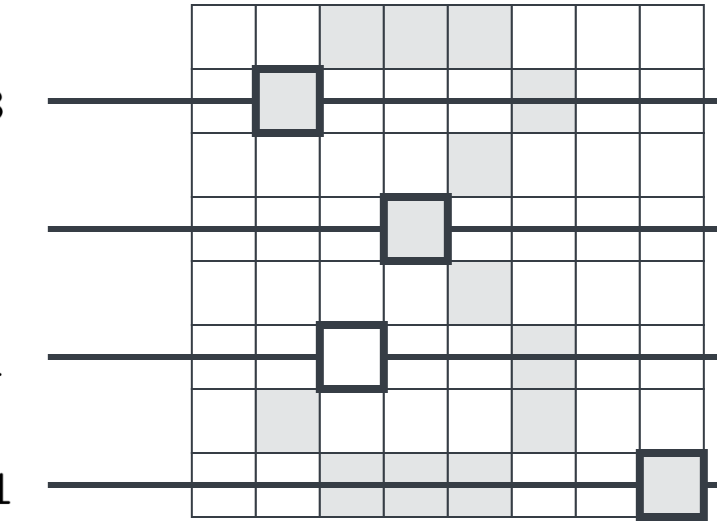
$P(C = 2) = 0.1$

$P(\text{on}|C = 2) = 0.8$

$P(\text{on}|C = 2) = 0.1$

$P(\text{off}|C = 2) = 0.1$

$P(\text{on}|C = 2) = 0.01$



$P(\text{features}, C = 3)$

$P(C = 3) = 0.1$

$P(\text{on}|C = 3) = 0.8$

$P(\text{on}|C = 3) = 0.9$

$P(\text{off}|C = 3) = 0.7$

$P(\text{on}|C = 3) = 0.0$

2 wins!!

Example: Overfitting

- Posteriors determined by *relative* probabilities (odds ratios):

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

```
south-west : inf
nation     : inf
morally    : inf
nicely     : inf
extent     : inf
seriously  : inf
...
```

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

```
screens    : inf
minute     : inf
guaranteed : inf
$205.00    : inf
delivery   : inf
signature  : inf
...
```

What went wrong here?

Generalization and Overfitting

- Relative frequency parameters will **overfit** the training data!
  - Just because we never saw a 3 with pixel (15,15) on during training doesn't mean we won't see it at test time
  - Unlikely that every occurrence of "minute" is 100% spam
  - Unlikely that every occurrence of "seriously" is 100% ham
  - What about all the words that don't occur in the training set at all?
  - In general, we can't go around giving unseen events zero probability

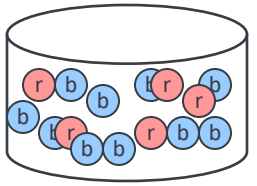
- As an extreme case, imagine using the entire email as the only feature
  - Would get the training data perfect (if deterministic labeling)
  - Wouldn't *generalize* at all
  - Just making the bag-of-words assumption gives us some generalization, but isn't enough

- To generalize better: we need to **smooth** or **regularize** the estimates.

Parameter Estimation

Parameter Estimation

- Estimating the distribution of a random variable
- **Elicitation**: ask a human (why is this hard?)
- **Empirically**: use training data (learning!)
  - E.g.: for each outcome x, look at the empirical rate of that value:



$$P_{\text{ML}}(x) = \frac{\text{count}(x)}{\text{total samples}}$$

$$P_{\text{ML}}(\text{r}) = 2/3$$

  - This is the estimate that maximizes the likelihood of the data.

$$L(x, \theta) = \prod_i P_\theta(x_i)$$

Smoothing (Regularization)

Maximum Likelihood?

- Relative frequencies are the maximum likelihood estimates.

$$\theta_{ML} = \arg\max_{\theta} P(\mathbf{X}|\theta)$$
$$= \arg\max_{\theta} \prod_{i} P_{\theta}(X_i)$$

$$\Rightarrow \quad P_{\text{ML}}(x) = \frac{\text{count}(x)}{\text{total samples}}$$

- Example: probability distribution of blue and red jellybeans
  - X = {Red, Red, Blue}
  - r = probability of picking a red jellybean
  - $P(X|r) = r^2(1-r)$
  - $P_{ML}(X) = \langle 2/3, 1/3 \rangle$

ML vs. MAP Learning

- Maximum Likelihood (ML)
  - Find parameters that maximize the prob of seeing the data D.
  - Easy to compute (*e.g.*, just counting)
  - Assumes uniform prior

$$\theta_{ML} = \arg\max_{\theta} P(\mathbf{X}|\theta)$$
$$= \arg\max_{\theta} \prod_i P_\theta(X_i)$$

- Prior: your belief before seeing any data
  - Uniform prior: all parameters equally likely

- Maximum a posteriori estimate
  - Maximize probability of parameters after seeing data D
  - Allows user to input additional domain knowledge
  - Better parameters when data is sparse
  - Reduces ML when infinite data

$$\theta_{MAP} = \arg\max_{\theta} P(\theta|\mathbf{X})$$
$$= \arg\max_{\theta} P(\mathbf{X}|\theta)P(\theta)/P(\mathbf{X})$$
$$= \arg\max_{\theta} P(\mathbf{X}|\theta)P(\theta)$$

Unseen Events

Laplace Smoothing

- Laplace's estimate:
  - Pretend you saw every outcome once more than you actually did.

$$P_{LAP}(x) = \frac{c(x)+1}{\sum_x [c(x)+1]}$$

$$= \frac{c(x)+1}{N+|X|}$$

$$P_{ML}(X) =$$

$$P_{LAP}(X) =$$

Laplace Smoothing

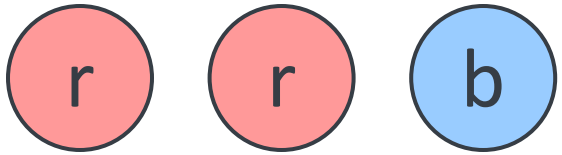
- Laplace's estimate (extended):
  - Pretend you saw every outcome k extra times

$$P_{LAP,k}(x) = \frac{c(x) + k}{N + k|X|}$$

  - What's Laplace with k = 0?
  - k is the **strength** of the prior

- Laplace for conditionals:
  - Smooth each condition independently:

$$P_{LAP,k}(x|y) = \frac{c(x,y) + k}{c(y) + k|X|}$$



$$P_{LAP,0}(X) =$$

$$P_{LAP,1}(X) =$$

$$P_{LAP,100}(X) =$$

Estimation: Linear Interpolation

- In practice, Laplace often performs poorly for P(X|Y):
  - When |X| is very large
  - When |Y| is very large

- Another option: linear interpolation
  - Also get the empirical P(X) from the data
  - Make sure the estimate of P(X|Y) isn't too different from the empirical P(X)

$$P_{LIN}(x|y) = \alpha \hat{P}(x|y) + (1.0 - \alpha)\hat{P}(x)$$

  - What if α is 0? 1?

Real NB: Smoothing

- For real classification problems, smoothing is critical.
- New odds ratios:

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

```
helvetica : 11.4
seems     : 10.8
group     : 10.2
ago       :  8.4
areas     :  8.3
...
```

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

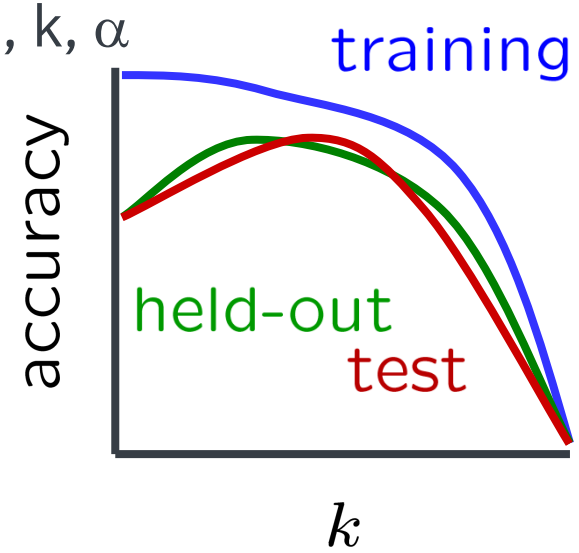
```
verdana : 28.8
Credit  : 28.4
ORDER   : 27.2
<FONT>  : 26.9
money   : 26.5
...
```

Do these make more sense?

Tuning

Tuning on Held-Out Data

- Now we've got two kinds of unknowns
  - Parameters: the probabilities P(X|Y), P(Y)
  - Hyperparameters: e.g., the amount / type of smoothing to do, k, α



- What should we learn where?
  - Learn parameters from training data
  - Tune hyperparameters on different data
    - Why?
  - For each value of the hyperparameters, train and test on the held-out data
  - Choose the best value and do a final test on the test data

Features

Errors, and What to Do

- Examples of errors

```
Dear GlobalSCAPE Customer,

GlobalSCAPE has partnered with ScanSoft to offer you the
latest version of OmniPage Pro, for just $99.99* - the regular
list price is $499! The most common question we've received
about this offer is - Is this genuine? We would like to assure
you that this offer is authorized by ScanSoft, is genuine and
valid. You can get the . . .
```

```
. . . To receive your $30 Amazon.com promotional certificate,
click through to

  http://www.amazon.com/apparel

and see the prominent link for the $30 offer. All details are
there. We hope you enjoyed receiving this message. However, if
you'd rather not receive future e-mails announcing new store
launches, please click . . .
```

What to Do About Errors?

- Need more features– words aren't enough!
  - Have you emailed the sender before?
  - Have 1K other people just gotten the same email?
  - Is the sending information consistent?
  - Is the email in ALL CAPS?
  - Do inline URLs point where they say they point?
  - Does the email address you by (your) name?

- Can add these information sources as new variables in the NB model

Baselines

- First step: get a **baseline**
  - Baselines are very simple “straw man” procedures
  - Help determine how hard the task is
  - Help know what a “good” accuracy is

- Weak baseline: most frequent label classifier
  - Gives all test instances whatever label was most common in the training set
  - E.g., for spam filtering, might label everything as ham
  - Accuracy might be very high if the problem is skewed
  - E.g., calling everything “ham” gets 66%, so a classifier that gets 70% isn’t very good…

- For real research, usually use previous work as a (strong) baseline

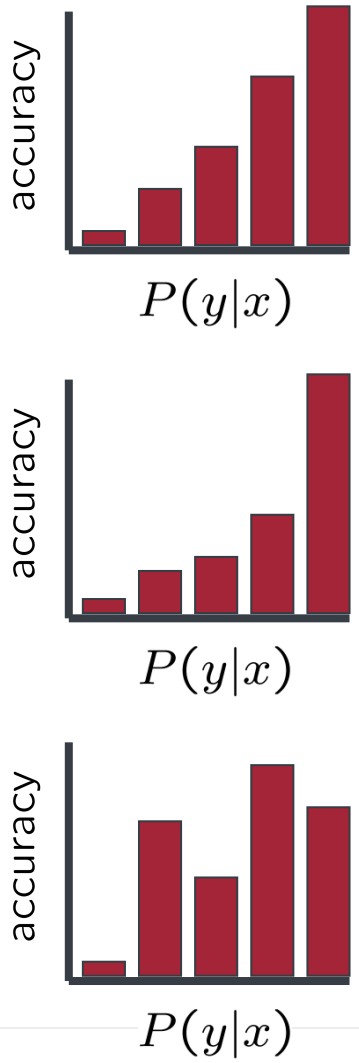
Confidences from a Classifier

- The **confidence** of a probabilistic classifier:
  - Posterior over the top label

$$\text{confidence}(x) = \max_y P(y|x)$$

  - Represents how sure the classifier is of the classification
  - Any probabilistic model will have confidences
  - No guarantee, confidence is correct

- Calibration
  - Weak calibration: higher confidences mean higher accuracy
  - Strong calibration: confidence predicts accuracy rate
  - What's the value of calibration?



Summary

- Bayes rule lets us do diagnostic queries with causal probabilities.
- The naïve Bayes assumption takes all features to be independent given the class label.
- We can build classifiers out of a naïve Bayes model using training data.
- Smoothing estimates is important in real systems.
- Classifier confidences are useful, when you can get them.

STEVENS
INSTITUTE OF TECHNOLOGY
1870

Questions?

Acknowledgement

Fahiem Bacchus, University of Toronto
Dan Klein, UC Berkeley
Kate Larson, University of Waterloo