# CIS 519/419
# Applied Machine Learning

www.seas.upenn.edu/~cis519

Dan Roth

danroth@seas.upenn.edu

http://www.cis.upenn.edu/~danroth/

461C, 3401 Walnut

Slides were created by Dan Roth (for CIS519/419 at Penn or CS446 at UIUC), Eric Eaton for CIS519/419 at Penn, or from other authors who have made their ML slides available.

# Administration (1)

- **Surveys:**
  - **Please do it.**
  - **If 80% of the students complete it, we'll give extra credit!**

- **Projects:**

- Come to my office hours at least once to discuss the project.
- **Posters** for the projects will be presented on the last meeting of the class, December 10, 12:00-1:30.
- **Final reports** will only be due after the Final exam, on December 18
  - Specific instructions are on the web page and will be sent also on Piazza.

- HW5: Will be released today.
  - It will be much shorted than earlier HWs.
  - Intended to prepare you for the exam.

# Administration (2)

- Exam:
  - The exam will take place on the originally assigned date, 12/17.
    - CHEM 102, 12-2pm
    - Structured similarly to the midterm.
    - 120 minutes; closed books.
  - What is covered:
    - Cumulative!
    - Slightly more focus on the material covered after the previous mid-term.
    - However, notice that the ideas in this class are cumulative!!
    - Everything that we present in class and in the homework assignments
    - Material that is in the slides but is not discussed in class is not part of the material required for the exam.
      - Example 1: We talked about Boosting. But not about boosting the confidence.
      - Example 2: We talked about multiclass classification: OvA, AvA, but not Error Correcting codes, and not about constraint classification (in the slides).
    - We will give practice exams. HW5 will also serve as preparation.

# So far…

- Bayesian Learning
  - What does it mean to be *Bayesian?*
- Naïve Bayes
  - Independence assumptions
- EM Algorithm
  - Learning with hidden variables
- Today:
  - Representing arbitrary probability distributions
  - Inference
    - Exact inference; Approximate inference
  - Learning Representations of Probability Distributions

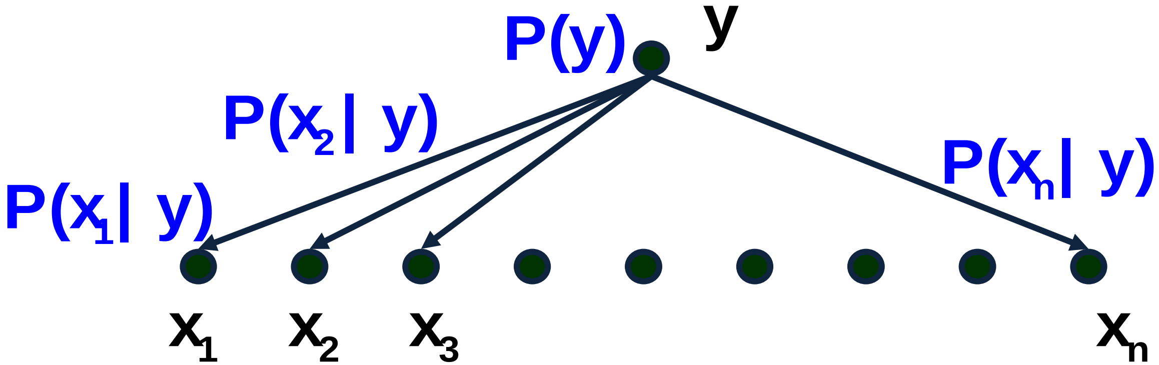
# Unsupervised Learning

- We get as input (n+1) tuples: ($X_1$, $X_2$, … $X_n$, $X_{n+1}$)
- There is no notion of a class variable or a label.
- After seeing a few examples, we would like to know something about the domain:
  - correlations between variables, probability of certain events, etc.
- We want to learn the most likely model that generated the data

- Sometimes called density estimation.

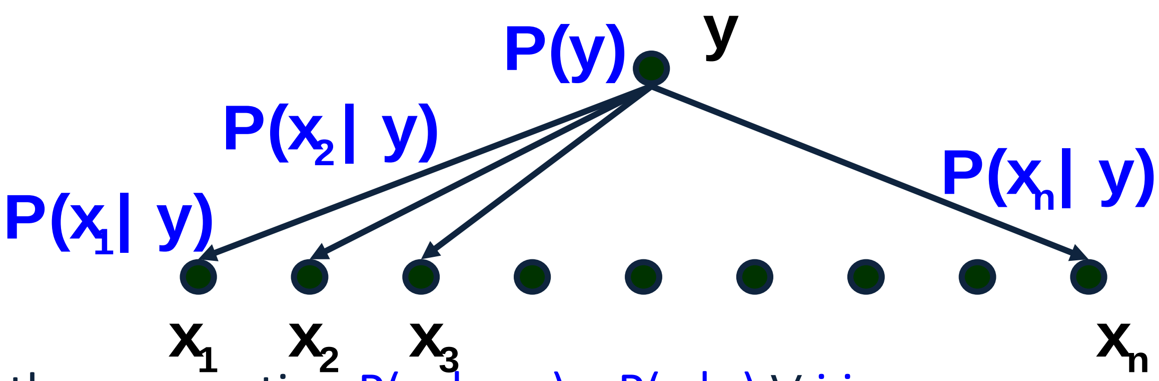
# Simple Distributions

- In general, the problem is very hard. But, under some assumptions on the distribution we have shown that we can do it. (exercise: show it's the most likely distribution)



- Assumptions: (conditional independence given y)
  - $P(x_i \mid x_j, y) = P(x_i \mid y) \; \forall \; i,j$
- Can these (strong) assumptions be relaxed ?
- Can we learn more general probability distributions ?
  - (These are essential in many applications: language, vision.)

# Simple Distributions



- Under the assumption $P(x_i \mid x_j, y) = P(x_i \mid y)\ \forall\ i,j$ we can compute the joint probability distribution on the n+1 variables

  $$P(y, x_1, x_2, \ldots x_n) = p(y)\prod_1^n P(x_i \mid y)$$

- Therefore, we can compute the probability of any event:
- $P(x_1 = 0, x_2 = 0, y = 1) = \sum_{\{b_i \in \{0,1\}\}} P(y=1, x_1=0, x_2=0, x_3=b_3, x_4=b_4, \ldots, x_n=b_n)$
- More efficiently (directly from the independence assumption):

  $$P(x_1 = 0, x_2 = 0, y = 1) = P(x_1=0, x_2=0 \mid y=1)\ p(y=1) =$$

  $$= P(x_1=0 \mid y=1)\ P(x_2=0 \mid y=1)\ p(y=1)$$

- We can compute the probability of any event or conditional event over the n+1 variables.

# Representing Probability Distribution

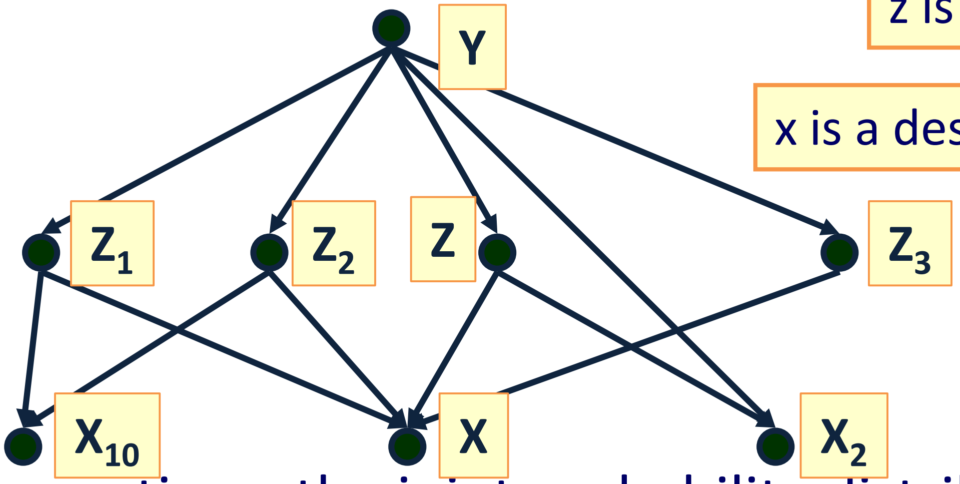
- Goal: To represent all joint probability distributions over a set of random variables $X_1, X_2, \ldots., X_n$
- There are many ways to represent distributions.
- A table, listing the probability of each instance in $\{0,1\}^n$
  - We will need $2^n$-1 numbers
- What can we do? Make Independence Assumptions

- Multi-linear polynomials
  - Multinomials over variables
- Bayesian Networks
  - Directed acyclic graphs
- Markov Networks
  - Undirected graphs

# Graphical Models of Probability Distributions

- Bayesian Networks represent the joint probability distribution over a set of variables.
- Independence Assumption: ∀ x, x is independent of its non-descendants given its parents

This is a theorem. To prove it, order the nodes from leaves up, and use the product rule.
The terms are called CPTs (Conditional Probability tables) and they completely define the probability distribution.



z is a parent of x

x is a descendant of y

- With these conventions, the joint probability distribution is given by:

$$P(y, x_1, x_2, \ldots x_n) = p(y) \prod_i P(x_i \mid Parents(x_i))$$

# Bayesian Network

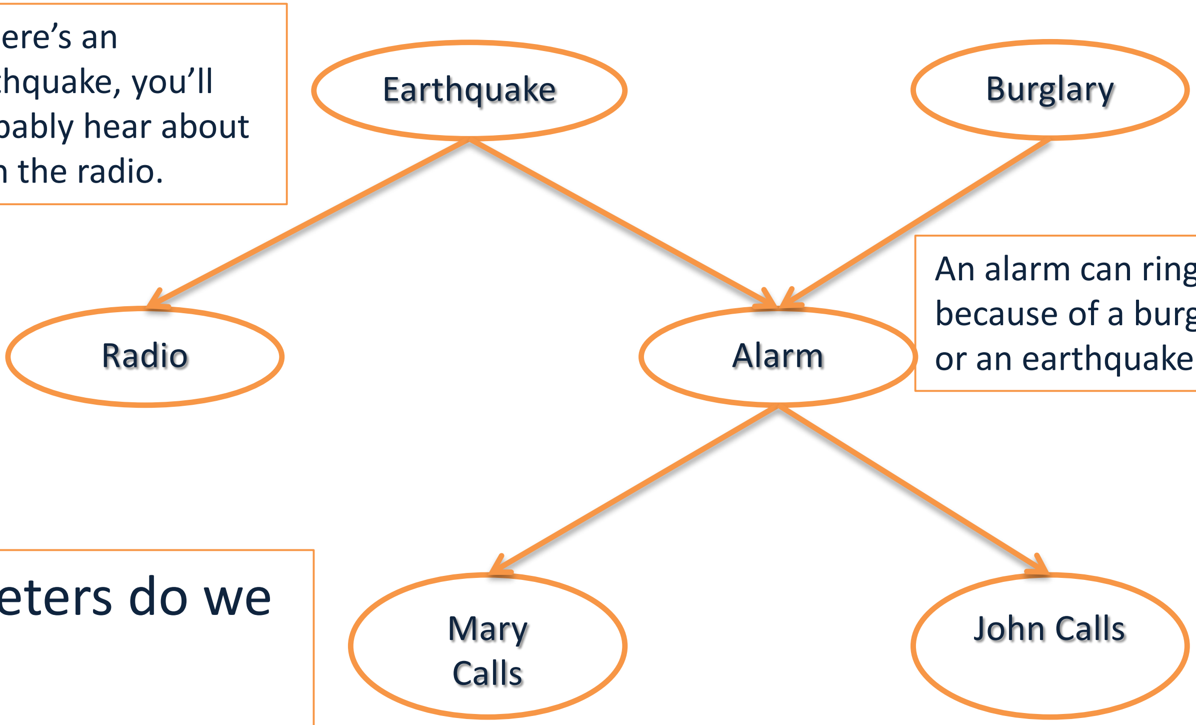
- Semantics of the DAG
  - Nodes are random variables
  - Edges represent causal influences
  - Each node is associated with a conditional probability distribution
- Two equivalent viewpoints
  - A data structure that represents the joint distribution compactly
  - A representation for a set of conditional independence assumptions about a distribution

# Bayesian Network: Example

The burglar alarm in your house rings when there is a burglary or an earthquake. An earthquake will be reported on the radio. If an alarm rings and your neighbors hear it, they will call you.

What are the random variables?

# Bayesian Network: Example



If there's an earthquake, you'll probably hear about it on the radio.

An alarm can ring because of a burglary or an earthquake.

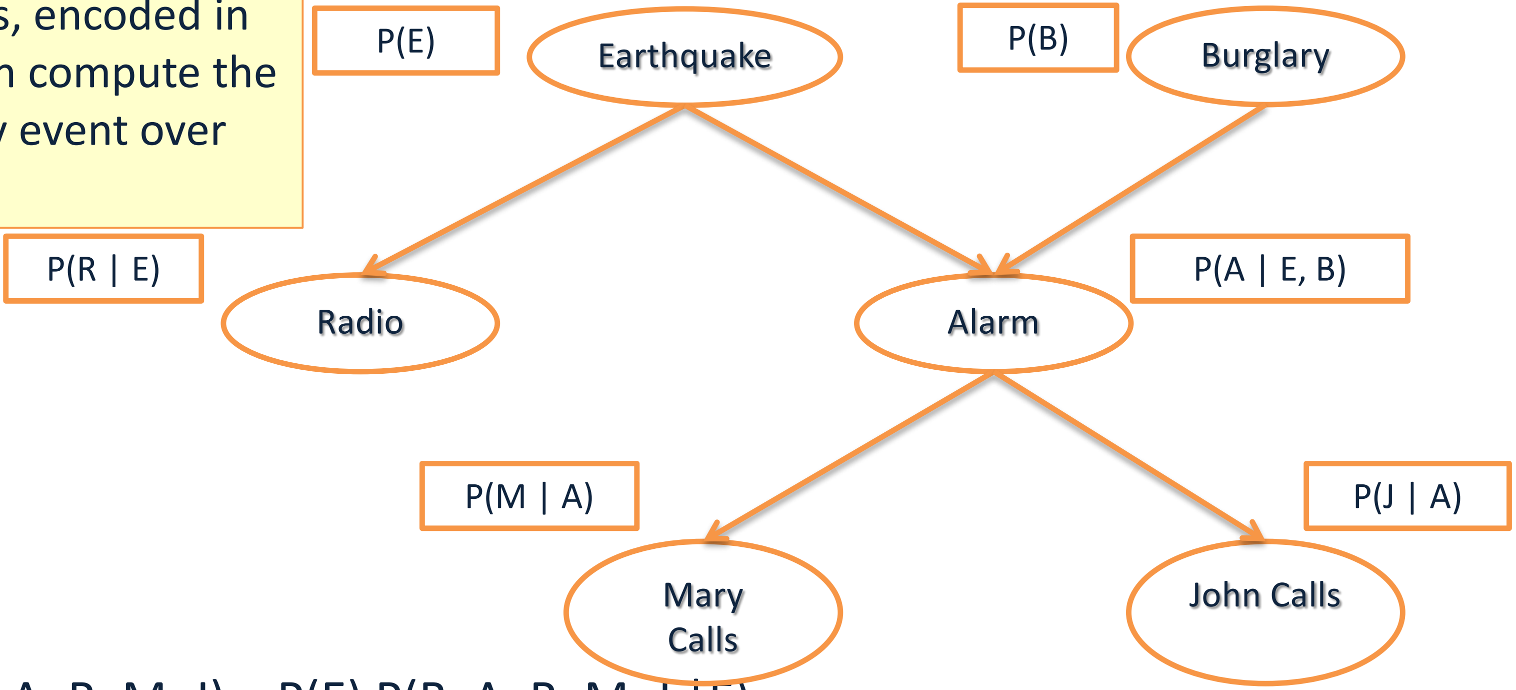
If your neighbors hear an alarm, they will call you.

How many parameters do we have?

How many would we have if we had to store the entire joint?

# Bayesian Network: Example

With these probabilities, (and assumptions, encoded in the graph) we can compute the probability of any event over these variables.



- $P(E, B, A, R, M, J) = P(E)\ P(B, A, R, M, J \mid E) =$

  $= P(E)\ P(B)\ P(A, R, M, J \mid E, B) =$

  $= P(E)\ P(B)\ P(R \mid E, B)\ P(M, J, A \mid E, B)$

  $= P(E)\ P(B)\ P(R \mid E)\ P(M, J \mid A, E, B)\ P(A \mid , E, B)$

  $= P(E)\ P(B)\ )\ P(R \mid E)\ P(M \mid A)\ P(J \mid A)\ P(A \mid E, B)$

# Computational Problems

- Learning the structure of the Bayes net
  - (What would be the guiding principle?)
- Learning the parameters
  - Supervised? Unsupervised?
- Inference:
  - Computing the probability of an event: [#P Complete, Roth'93, '96]
    - Given structure and parameters
    - Given an observation E, what is the probability of assignment Y?
      - **P(R=off, A=off | E=e) =?** (E, Y are sets of instantiated variables)
  - Most likely explanation (Maximum A Posteriori assignment, MAP, MPE) [NP-Hard; Shimony'94]
    - Given structure and parameters
    - Given an observation E, what is the most likely assignment to Y?
    - Argmax$_y$ **P(Y=y | E=e) (Say, Y = (R, A))**
    - (E, Y are sets of instantiated variables)

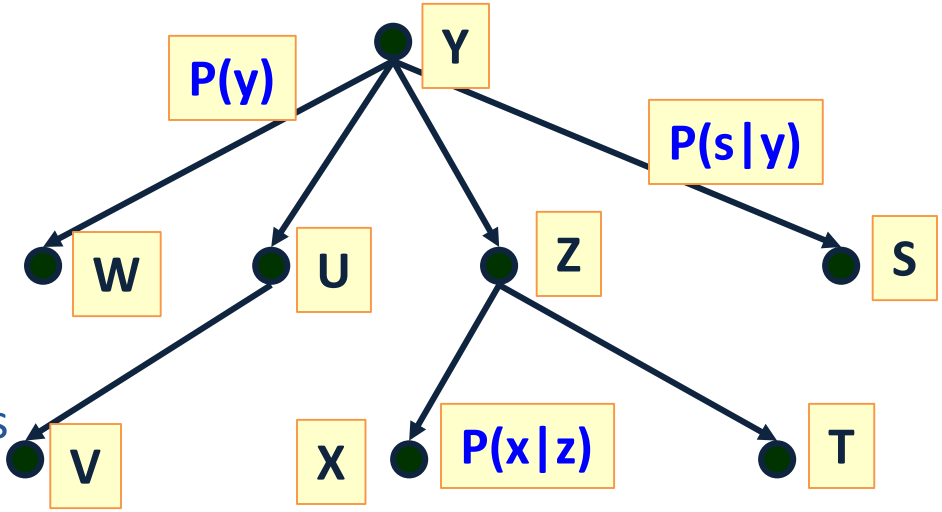
# Inference

- Inference in Bayesian Networks is generally intractable in the worst case

- Two broad approaches for inference
  - Exact inference
    - Eg. Variable Elimination
  - Approximate inference
    - Eg. Gibbs sampling

# Tree Dependent Distributions

- Directed Acyclic graph
  - Each node has at most one parent
- Independence Assumption:
  - x is independent of its non-descendants given its parents
- (x is independent of other nodes give z; v is independent of w given u;)

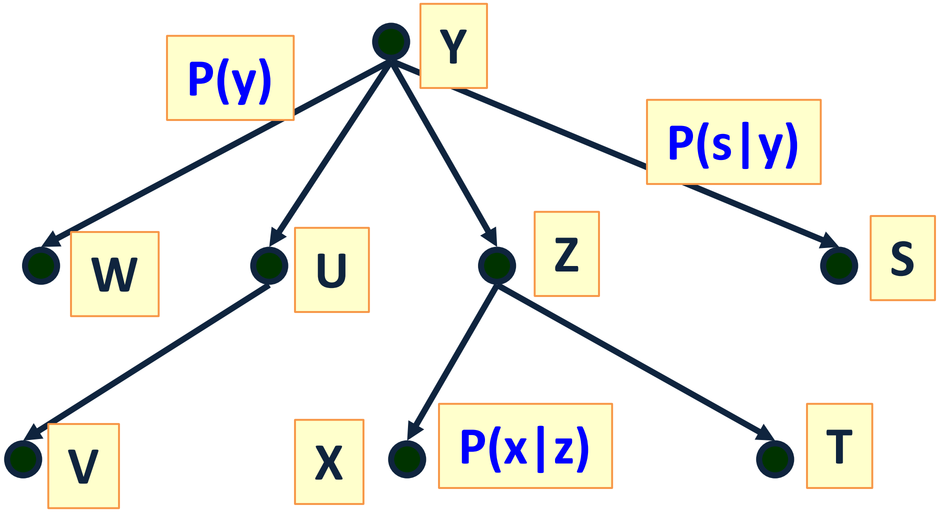


$$P(y, x_1, x_2, \ldots x_n) = p(y) \prod_i P(x_i \mid Parents(x_i))$$

- Need to know two numbers for each link: p(x|z), and a prior for the root p(y)

# Tree Dependent Distributions

- This is a generalization of naïve Bayes.
- Inference Problem:
  - Given the Tree with all the associated probabilities, evaluate the probability of an event p(x) ?



$$P(y, x_1, x_2, \ldots x_n) = p(y) \prod_i P(x_i \mid Parents(x_i))$$

- P(x=1) =

  = P(x=1|z=1)P(z=1) + P(x=1|z=0)P(z=0)

- Recursively, go up the tree:

  P(z=1) = P(z=1|y=1)P(y=1) + P(z=1|y=0)P(y=0)

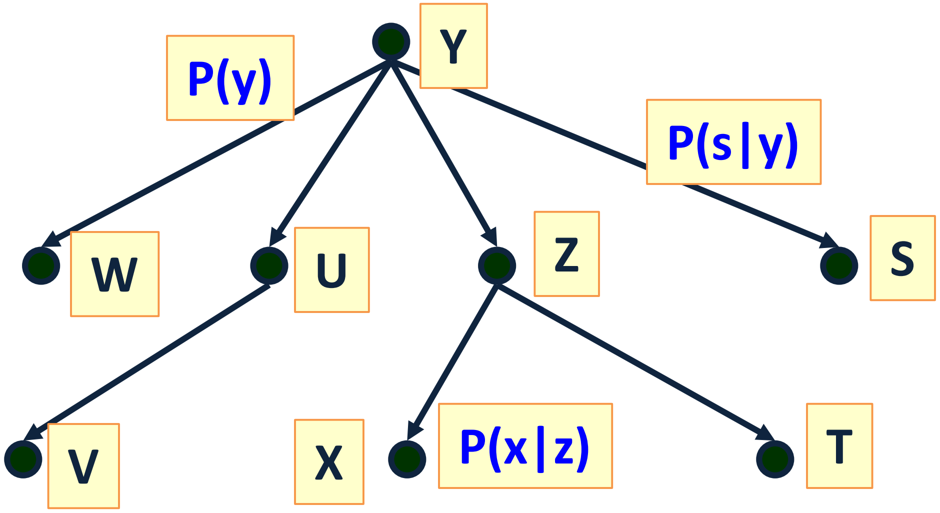
  P(z=0) = P(z=0|y=1)P(y=1) + P(z=0|y=0)P(y=0)

  Linear Time Algorithm

Now we have everything in terms of the CPTs (conditional probability tables)

# Tree Dependent Distributions

- This is a generalization of naïve Bayes.
- Inference Problem:
  - Given the Tree with all the associated probabilities, evaluate the probability of an event p(x,y) ?



$$P(y, x_1, x_2, \ldots x_n) = p(y) \prod_i P(x_i \mid Parents(x_i))$$

- $P(x=1,y=0) =$

  $= P(x=1|y=0)P(y=0)$

- Recursively, go up the tree along the path from x to y:

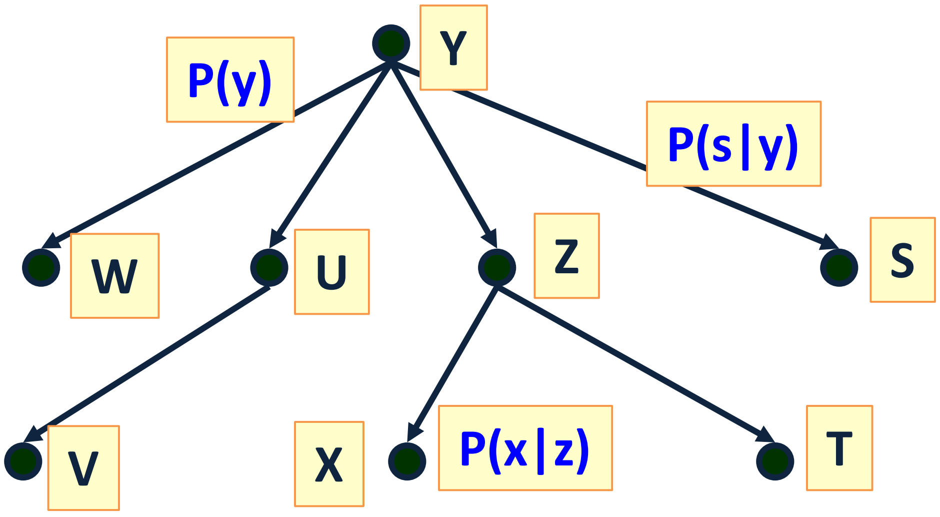
  $P(x=1|y=0) = \sum_{z=0,1} P(x=1|y=0, z)P(z|y=0) =$

  $= \sum_{z=0,1} P(x=1|z)P(z|y=0)$

Now we have everything in terms of the CPTs (conditional probability tables)

# Tree Dependent Distributions

- This is a generalization of naïve Bayes.
- Inference Problem:
  - Given the Tree with all the associated probabilities, evaluate the probability of an event p(x,u) ?
  - (No direct path from x to u)



$$P(y, x_1, x_2, \ldots x_n) = p(y) \prod_i P(x_i \mid \text{Parents}(x_i))$$

- $P(x=1,u=0) = P(x=1|u=0)P(u=0)$
- Let y be a parent of x and u (we always have one)

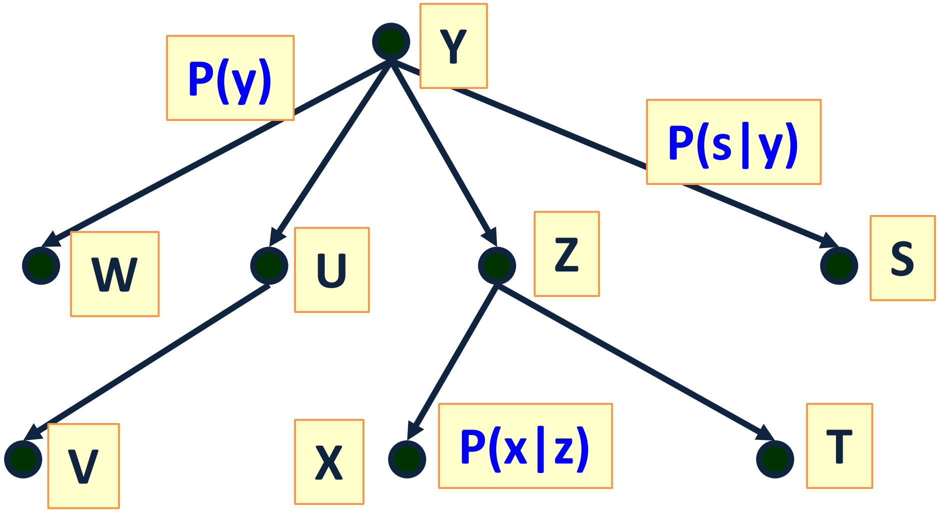
$P(x=1|u=0) = \sum_{y=0,1} P(x=1|u=0, y)P(y|u=0) =$

$= \sum_{y=0,1} P(x=1|y)P(y|u=0) =$

Now we have reduced it to cases we have seen

# Tree Dependent Distributions

- **Inference Problem:**



- Given the Tree with all the associated CPTs, we “showed” that we can evaluate the probability of all events efficiently.
- There are more efficient algorithms
- The idea was to show that the inference is this case is a simple application of Bayes rule and probability theory.

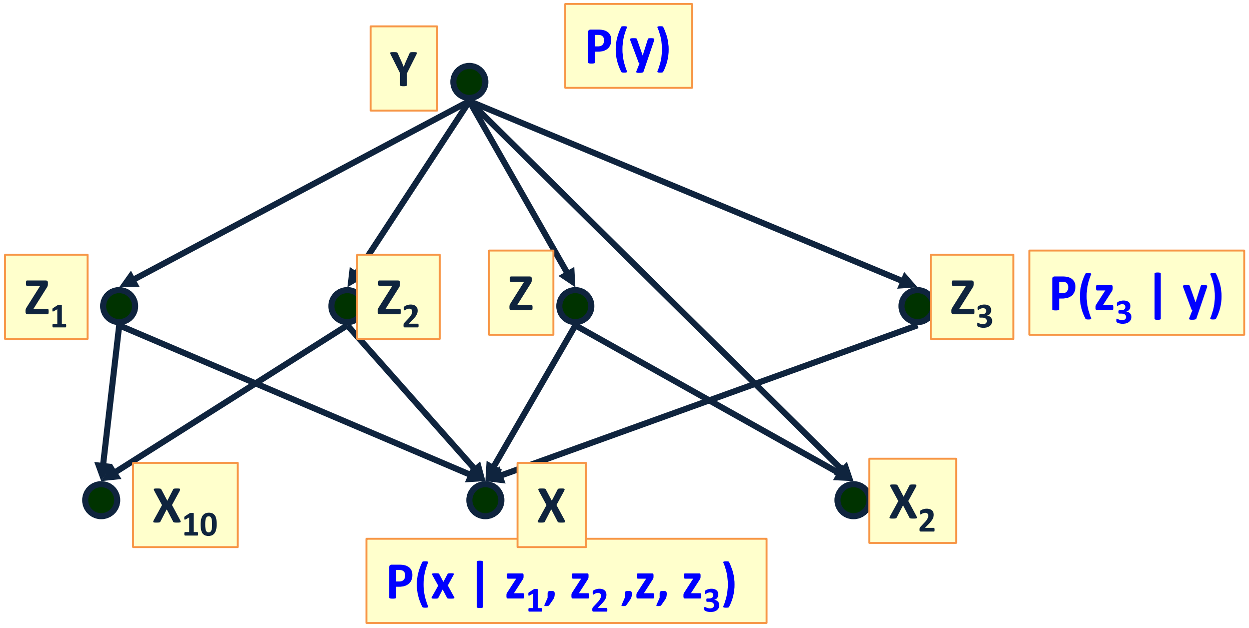
$$\mathbf{P(y, x_1, x_2, \ldots x_n) = p(y) \prod_i P(x_i \mid Parents(x_i))}$$

Things are not so simple in the general case, due to cycles; there are multiple ways to “get” from node A to B, and this has to be accounted for in Inference.

Skip Inference

# Graphical Models of Probability Distributions

- For general Bayesian Networks
  - The learning problem is hard
  - The inference problem (given the network, evaluate the probability of a given event) is hard (#P Complete)



$$P(y, x_1, x_2, \ldots x_n) = p(y) \prod_i P(x_i \mid \mathrm{Parents}(x_i))$$

# Variable Elimination

$$P(x_1,x_2,\ldots x_n)=\prod_i P(x_i \mid Parents(x_i))$$

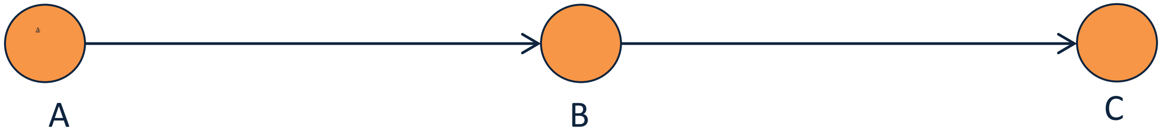
- Suppose the query is P($X_1$)

$$P(x_1)=\underline{\sum_{x_2,\cdots,x_n}}\ \underline{P(x_1,x_2,\cdots,x_n)}$$

$$P(x_1)=\underline{\sum_{x_2}\sum_{x_3}\cdots\sum_{x_n}}\ \underline{\prod_i P(x_i \mid Parents(x_i))}$$

- Key Intuition: Move irrelevant terms outside summation and cache intermediate results

# Variable Elimination: Example 1



- We want to compute P(C)

$$P(C) = \sum_A \sum_B P(A,B,C) = \sum_A \sum_B P(A)P(B|A)P(C|B)$$

$$= \sum_B P(C|B) \boxed{\sum_A P(A)P(B|A)} \rightarrow$$

Let's call this $f_A(B)$

$$= \sum_B P(C|B) f_A(B)$$

A has been (instantiated and) eliminated

- What have we saved with this procedure? How many multiplications and additions did we perform?

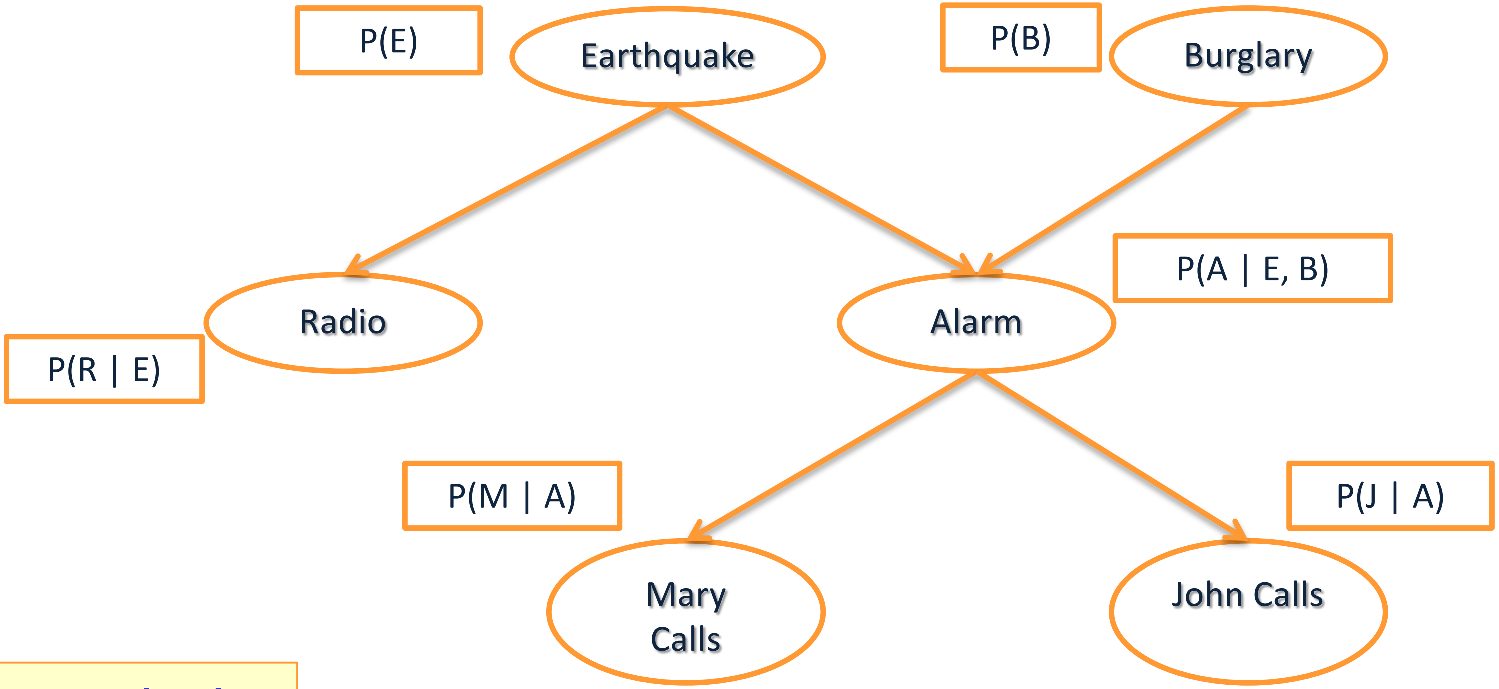
# Variable Elimination

- VE is a sequential procedure.

- Given an ordering of variables to eliminate
  - For each variable v that is not in the query
    - Replace it with a new function $f_v$
      - That is, marginalize v out

- The actual computation depends on the order
- What is the domain and range of $f_v$?
  - It need not be a probability distribution

# Variable Elimination: Example 2



What is P(M, J | B)?

$$P(E,B,A,R,M,J) = P(E \mid B,A,R,M,J)P(B,A,R,M,J)$$

$$= P(E)\cdot P(B)\cdot P(R \mid E)\cdot P(A \mid E,B)\cdot P(M \mid A)\cdot P(J \mid A)$$

# Variable Elimination: Example 2

$$P(E,B,A,R,M,J) = P(E)\cdot P(B)\cdot P(R\,|\,E)\cdot P(A\,|\,E,B)\cdot P(M\,|\,A)\cdot P(J\,|\,A)$$

$$P(M,J\,|\,B=true) = \frac{P(M,J,B=true)}{P(B=true)}$$

It is sufficient to compute the numerator and normalize

Assumptions (graph; joint representation)

$$P(M,J,B=true) = \sum_{E,A,R} P(E,B=true,A,R,M,J)$$

Elimination order R, A, E

$$= \sum_{E,A,R} P(E)\cdot P(B=true)\cdot P(R\,|\,E)\cdot P(A\,|\,E,B=true)\cdot P(M\,|\,A)\cdot P(J\,|\,A)$$

To eliminate R

$$f_R(E) = \sum_R P(R\,|\,E)$$

# Variable Elimination: Example 2

$$P(E,B,A,R,M,J) = P(E)\cdot P(B)\cdot P(R\mid E)\cdot P(A\mid E,B)\cdot P(M\mid A)\cdot P(J\mid A)$$

$$P(M,J\mid B=true) = \frac{P(M,J,B=true)}{P(B=true)}$$

It is sufficient to compute the numerator and normalize

---

$$P(M,J,B=true) = \sum_{E,A,R} P(E,B=true,A,R,M,J)$$

Elimination order A, E

$$= \sum_{E,A} P(E)\cdot P(B=true)\cdot P(A\mid E,B=true)\cdot P(M\mid A)\cdot P(J\mid A)\cdot \boxed{f_R(E)}$$

To eliminate A

$$f_A(E,M,J) = \sum_{A} P(A\mid E,B=true)\cdot P(M\mid A)\cdot P(J\mid A)$$

$$f_R(E) = \sum_{R} P(R\mid E)$$

# Variable Elimination: Example 2

$$P(E,B,A,R,M,J) = P(E)\cdot P(B)\cdot P(R\mid E)\cdot P(A\mid E,B)\cdot P(M\mid A)\cdot P(J\mid A)$$

$$P(M,J\mid B=true) = \frac{P(M,J,B=true)}{P(B=true)}$$

It is sufficient to compute the numerator and normalize

$$P(M,J,B=true) = \sum_{E,A,R} P(E,B=true,A,R,M,J)$$

$$= \sum_{E} P(E)\cdot P(B=true)\cdot f_A(E,M,J)\cdot f_R(E)$$

Finally eliminate E

Factors

$$f_R(E) = \sum_{R} P(R\mid E)$$

$$f_A(E,M,J) = \sum_{A} P(A\mid E,B=true)\cdot P(M\mid A)\cdot P(J\mid A)$$

# Variable Elimination

- The order in which variables are eliminated matters
  - In the previous example, what would happen if we eliminate E first?
    - The size of the factors would be larger
- Complexity of Variable Elimination
  - Exponential in the size of the factors
  - What about worst case?
    - The worst case is intractable

# Inference

- Exact Inference in Bayesian Networks is #P-hard
  - We can count the number of satisfying assignments for 3-SAT with a Bayesian Network

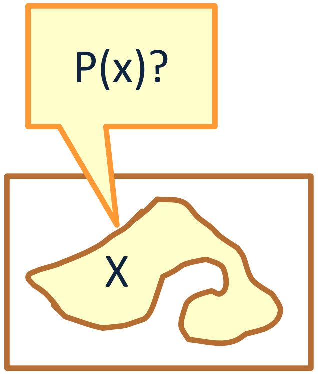
- Approximate inference
  - Eg. Gibbs sampling
  - Skip

# Approximate Inference

- Basic idea
  - If we had access to a set of examples from the joint distribution, we could just count.

$$E[f(x)] \approx \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)})$$

  - For inference, we generate instances from the joint and count
  - How do we generate instances?

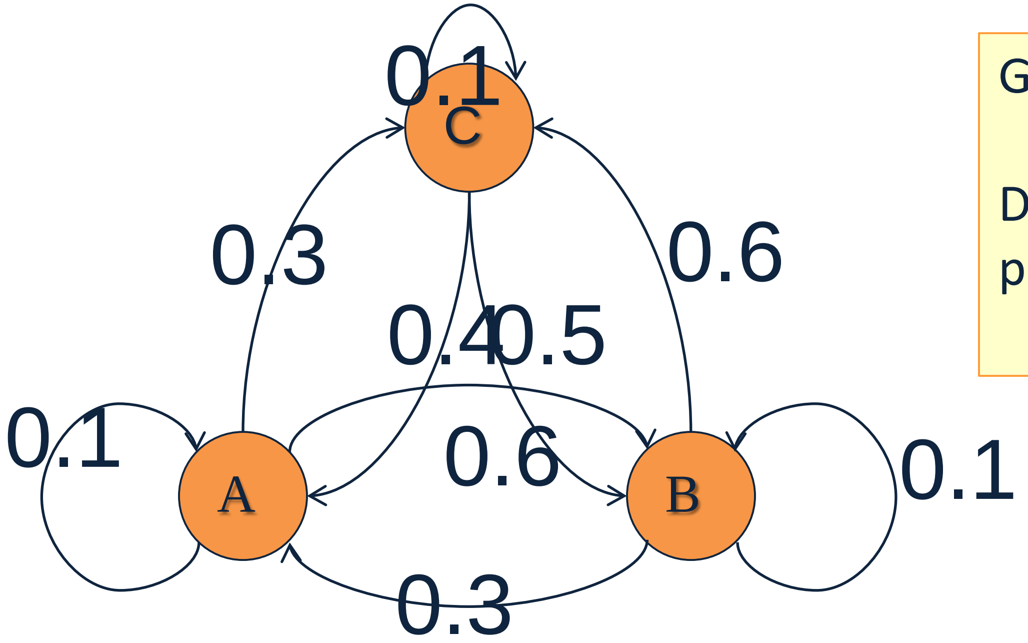


# Generating instances

- Sampling from the Bayesian Network
  - Conditional probabilities, that is, P(X|E)
  - Only generate instances that are consistent with E

- Problems?
  - How many samples? [Law of large numbers]
  - What if the evidence E is a very low probability event?
  - Skip

# Detour: Markov Chain Review



Generates a sequence of A,B,C

Defined by initial and transition probabilities $P(X_0)$ and $P(X_{t+1}=i \mid X_t=j)$

$P_{ij}$ : Time independent transition probability matrix

**Stationary Distributions:** A vector q is called a stationary distribution if

$$q_j = \sum_i q_i P_{ij}$$

$q_i$ : The probability of being in state i

If we sample from the Markov Chain repeatedly, the distribution over the states converges to the stationary distribution

# Markov Chain Monte Carlo

- Our goal: To sample from P(X| e)

- Overall idea:
  - The next sample is a function of the current sample
  - The samples can be thought of as coming from a Markov Chain whose stationary distribution is the distribution we want

- Can approximate any distribution

# Gibbs Sampling

- The simplest MCMC method to sample from $P(X=x_1x_2 \ldots x_n \mid e)$

- Creates a Markov Chain of samples as follows:
  - Initialize X randomly
  - At each time step, fix all random variables except one.
  - Sample that random variable from the corresponding conditional distribution

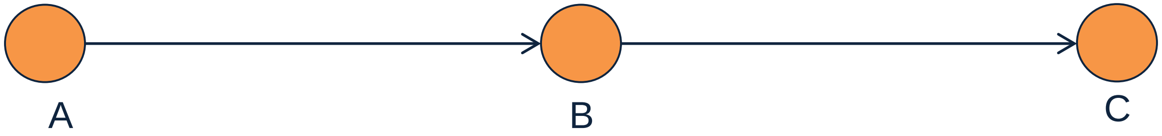
# Gibbs Sampling

- Algorithm:
  - Initialize X randomly
  - Iterate:
    - Pick a variable $X_i$ uniformly at random
    - Sample $x_i^{(t+1)}$ from $P(x_i \mid x_1^{(t)},\ldots,x_{i-1}^{(t)}, x_{i+1}^{(t),}\ldots, x_n^{(t)},e)$
    - $X_k^{(t+1)}=x_k^{(t+1)}$ for all other k
    - This is the next sample
- $X^{(1)},X^{(2)},\ldots X^{(t)}$ forms a Markov Chain
- Why is Gibbs Sampling easy for Bayes Nets?
  - $P(x_i \mid x_{-i}^{(t)},e)$ is “local”

# Gibbs Sampling: Big picture

- Given some conditional distribution we wish to compute, collect samples from the Markov Chain
- Typically, the chain is allowed to run for some time before collecting samples (burn in period)
    - So that the chain settles into the stationary distribution
- Using the samples, we approximate the posterior by counting

# Gibbs Sampling Example 1

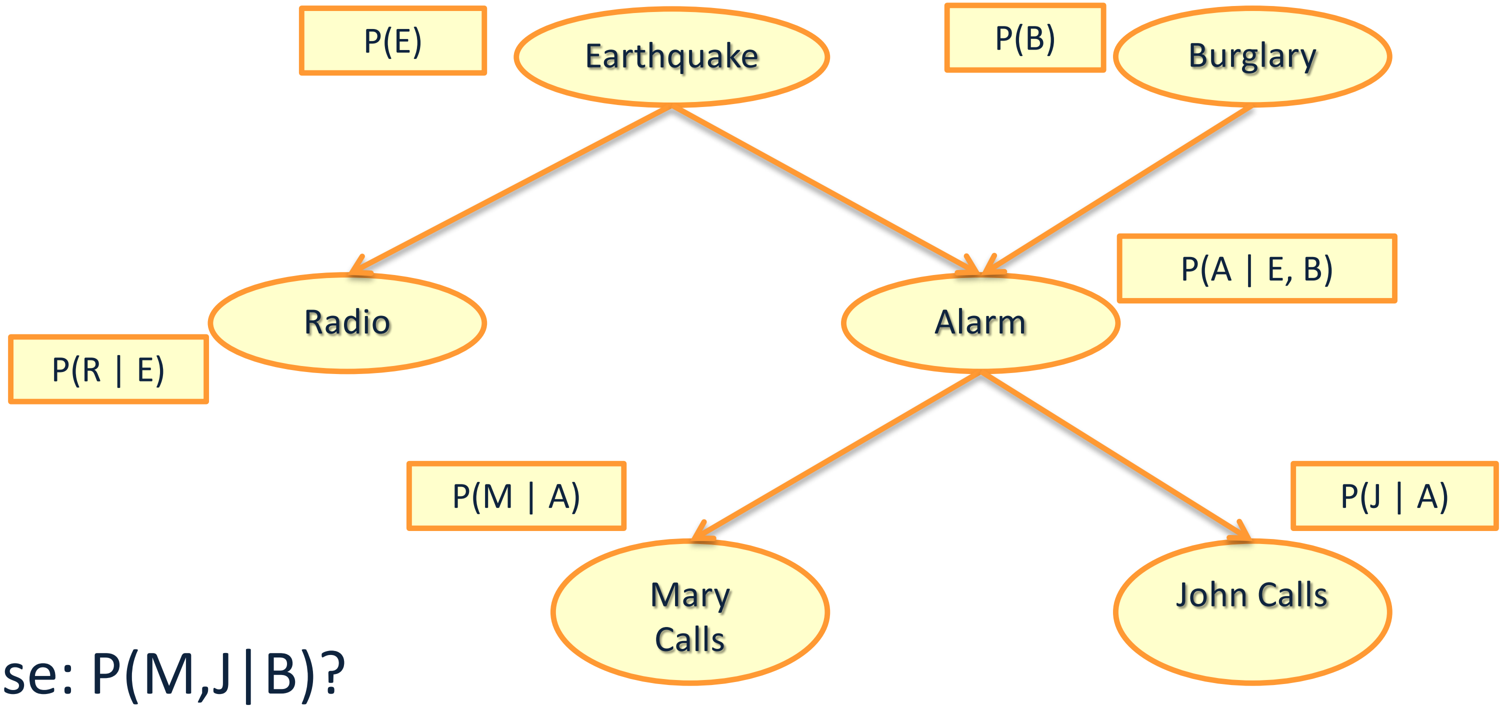


We want to compute P(C):

Suppose, after burn in, the Markov Chain is at A=true, B=false, C= false

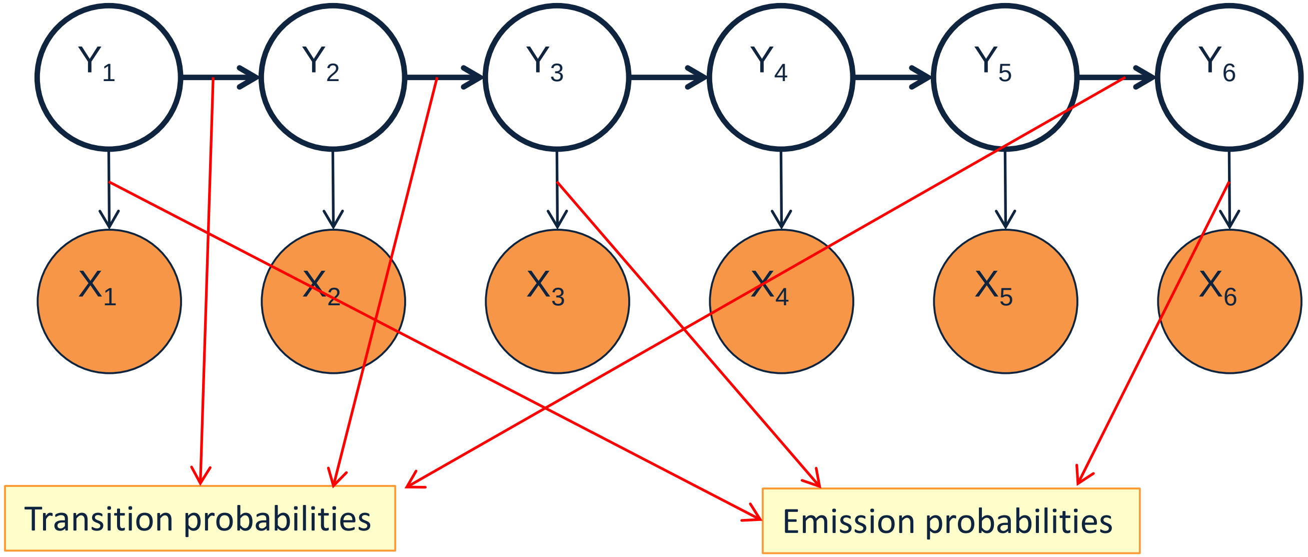
1. Pick a variable → B
2. Draw the new value of B from
   - P(B | A=true, C= false) = P(B | A=true)
   - Suppose $B^{new}$ = true
3. Our new sample is A=true, B = true, C = false
4. Repeat

# Gibbs Sampling Example 2



- Exercise: P(M,J|B)?

# Example: Hidden Markov Model



A Bayesian Network with a specific structure.
Xs are called the observations and Ys are the hidden states

Useful for sequence tagging tasks – part of speech, modeling temporal structure, speech recognition, etc

# HMM: Computational Problems

- Probability of an observation given an HMM
  - P(X| parameters): Dynamic Programming

- Finding the best hidden states for a given sequence
  - P(Y | X, parameters): Dynamic Programming

- Learning the parameters from observations
  - EM

# Gibbs Sampling for HMM

- Goal:Computing P(y|x)

- Initialize the Ys randomly
- Iterate:
  - Pick a random $Y_i$
  - Draw $Y_i^t$ from $P(Y_i|\ Y_{i-1},Y_{i+1},X_i)$

Only these variables are needed because they form the Markov blanket of $Y_i$.

- Compute the probability using counts after the burn in period

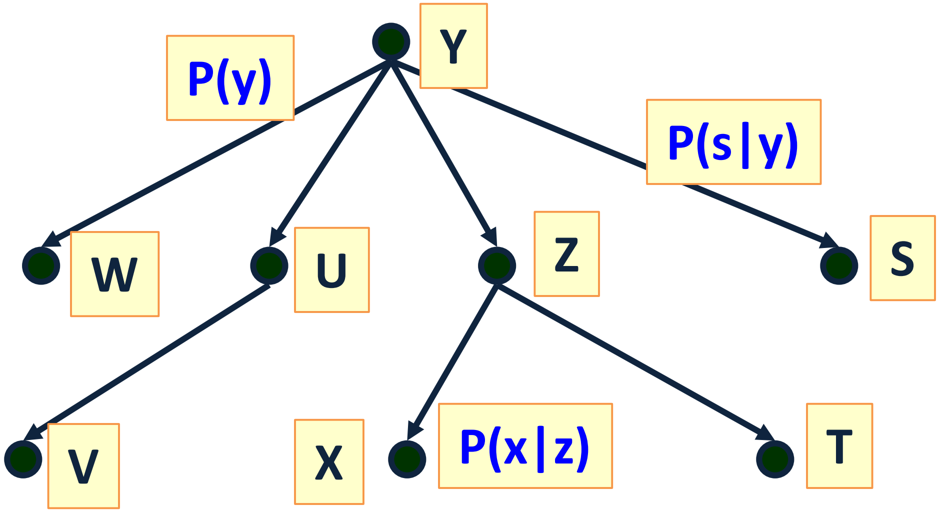
Gibbs sampling allows us to introduce priors on the emission and transition probabilities.

# Bayesian Networks

- Bayesian Networks
  - Compact representation probability distributions
  - Universal: Can represent all distributions
    - In the worst case, every random variable will be connected to all others
- Inference
  - Inference is hard in the worst case
    - Exact inference is #P-hard, approximate inference is NP-hard [Roth93,96]
    - Inference for Trees is efficient
    - General exact Inference: Variable Elimination
- Learning?

# Tree Dependent Distributions

- **Learning Problem:**



- Given data (n tuples) assumed to be sampled from a tree-dependent distribution
  - What does that mean?
  - Generative model

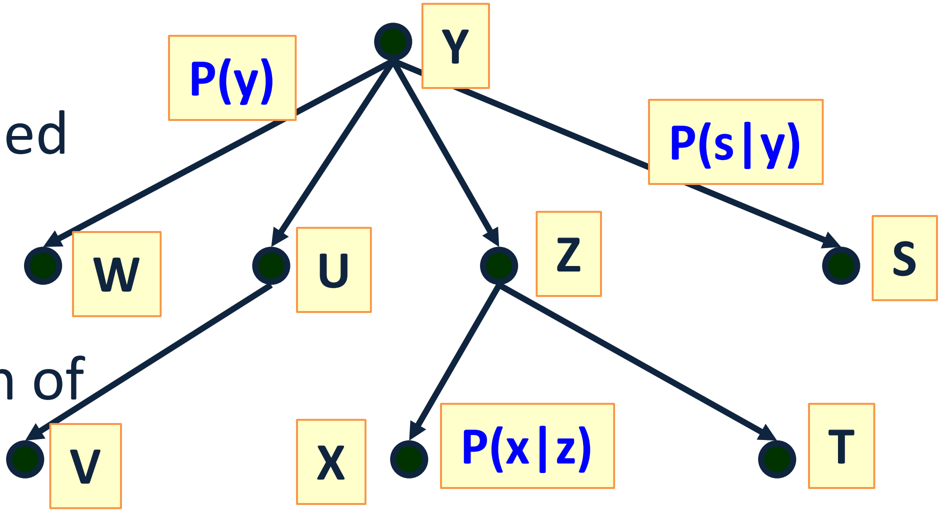
$$P(y, x_1, x_2, \ldots x_n) = p(y) \prod_i P(x_i \mid Parents(x_i))$$

- Find the tree representation of the distribution.
  - What does that mean?
- Among all trees, find the most likely one, given the data:

**P(T|D) = P(D|T) P(T)/P(D)**

# Tree Dependent Distributions

- **Learning Problem:**
- Given data (n tuples) assumed to be sampled from a tree-dependent distribution
- Find the tree representation of the distribution.



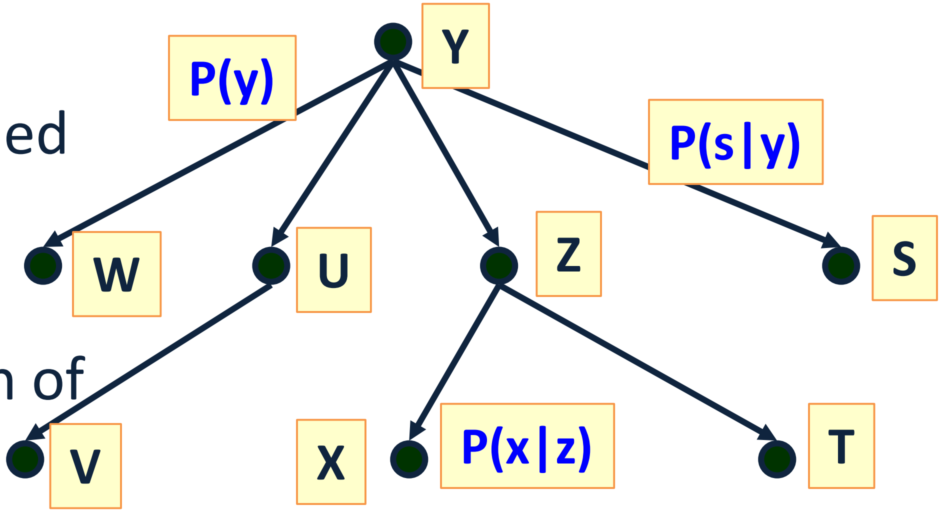
- Assuming uniform prior on trees, the Maximum Likelihood approach is to maximize P(D|T),

$$T_{ML} = \text{argmax}_T\, P(D|T) = \text{argmax}_T \prod_{\{x\}} P_T\,(x_1, x_2, \ldots x_n)$$

- Now we can see why we had to solve the inference problem first; it is required for learning.

# Tree Dependent Distributions

- **Learning Problem:**
- Given data (n tuples) assumed to be sampled from a tree-dependent distribution
- Find the tree representation of the distribution.



- Assuming uniform prior on trees, the Maximum Likelihood approach is to maximize P(D|T),

$$T_{ML} = \text{argmax}_T\, P(D|T) = \text{argmax}_T \prod_{\{x\}} P_T(x_1, x_2, \ldots x_n) =$$

$$= \text{argmax}_T \prod_{\{x\}} P_T(x_i|\text{Parents}(x_i))$$

Try this for naïve Bayes

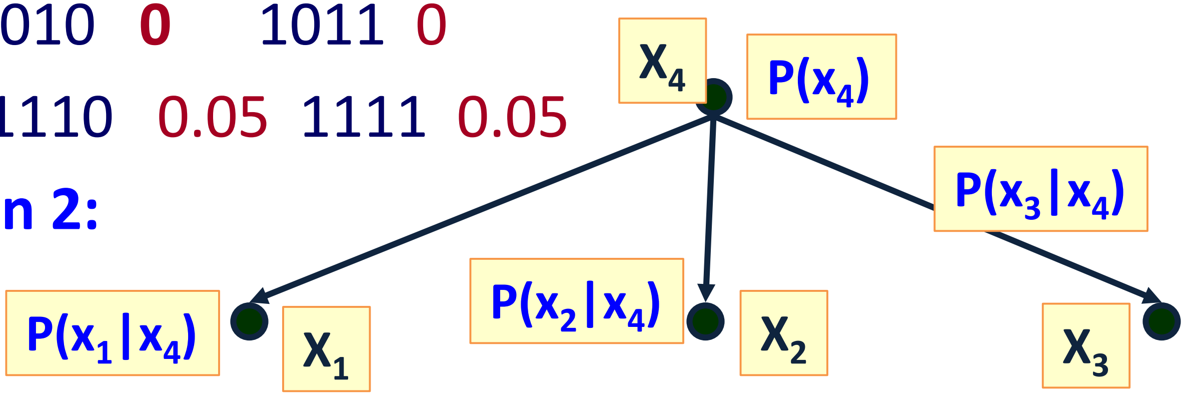
# Example: Learning Distributions

Are these representations of the same distribution? Given a sample, which of these generated it?

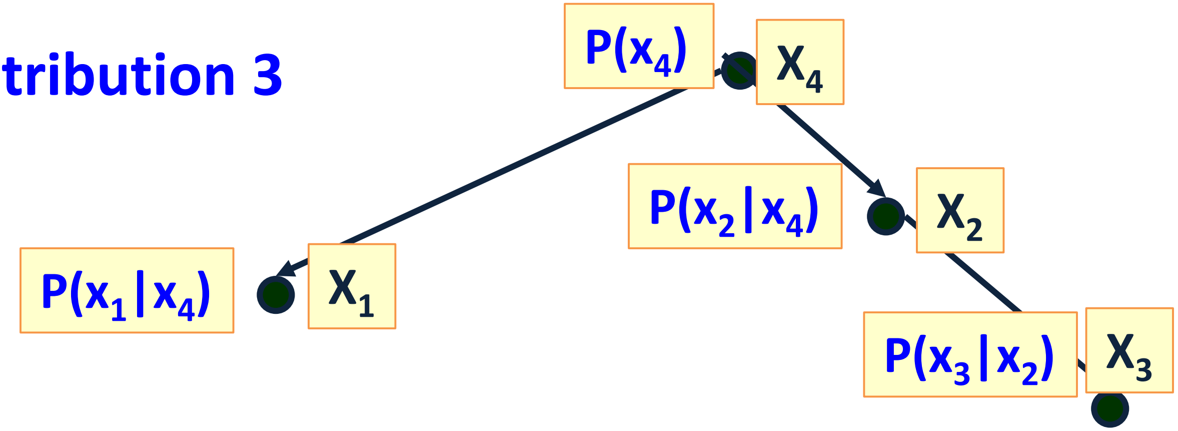
- **Probability Distribution 1:**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0000 | 0.1 | 0001 | 0.1 | 0010 | 0.1 | 0011 | 0.1 |
| 0100 | 0.1 | 0101 | 0.1 | 0110 | 0.1 | 0111 | 0.1 |
| 1000 | 0 | 1001 | 0 | 1010 | **0** | 1011 | 0 |
| 1100 | 0.05 | 1101 | 0.05 | 1110 | 0.05 | 1111 | 0.05 |

- **Probability Distribution 2:**



- **Probability Distribution 3**



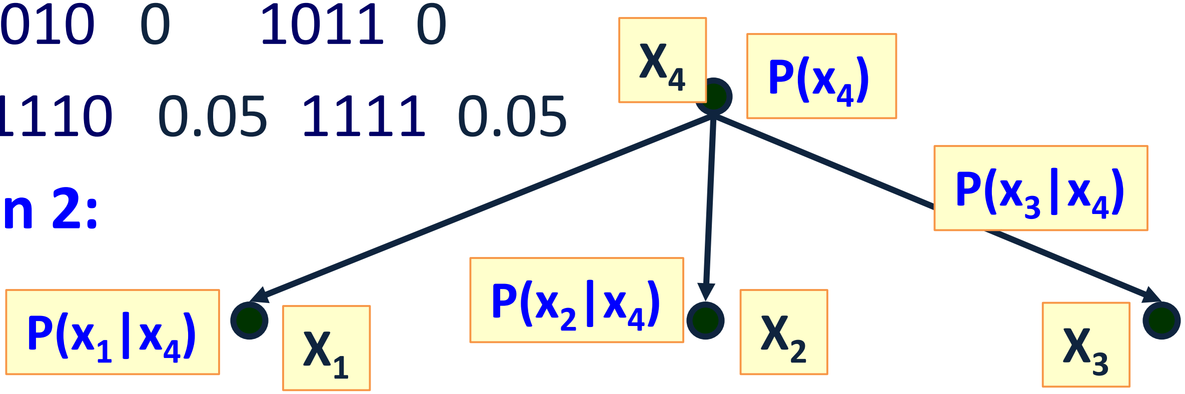
# Example: Learning Distributions

We are given 3 data points: 1011; 1001; 0100
Which one is the target distribution?

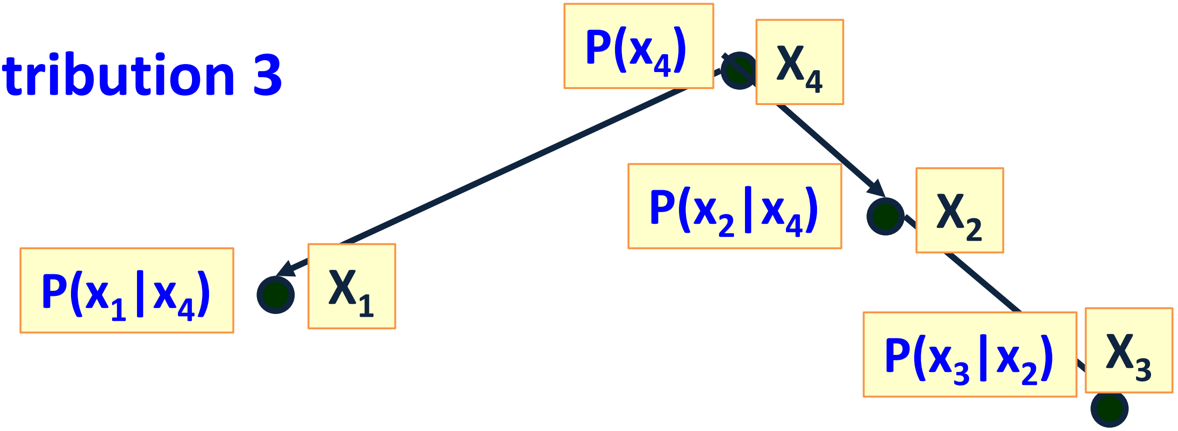
- **Probability Distribution 1:**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0000 | 0.1 | 0001 | 0.1 | 0010 | 0.1 | 0011 | 0.1 |
| 0100 | 0.1 | 0101 | 0.1 | 0110 | 0.1 | 0111 | 0.1 |
| 1000 | 0 | 1001 | 0 | 1010 | 0 | 1011 | 0 |
| 1100 | 0.05 | 1101 | 0.05 | 1110 | 0.05 | 1111 | 0.05 |

- **Probability Distribution 2:**



- **Probability Distribution 3**



# Example: Learning Distributions

We are given 3 data points: 1011; 1001; 0100 Which one is the target distribution?

- **Probability Distribution 1:**

|  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
| 0000 | 0.1 | 0001 | 0.1 | 0010 | 0.1 | 0011 | 0.1 |
| 0100 | 0.1 | 0101 | 0.1 | 0110 | 0.1 | 0111 | 0.1 |
| 1000 | 0 | 1001 | 0 | 1010 | 0 | 1011 | 0 |
| 1100 | 0.05 | 1101 | 0.05 | 1110 | 0.05 | 1111 | 0.05 |

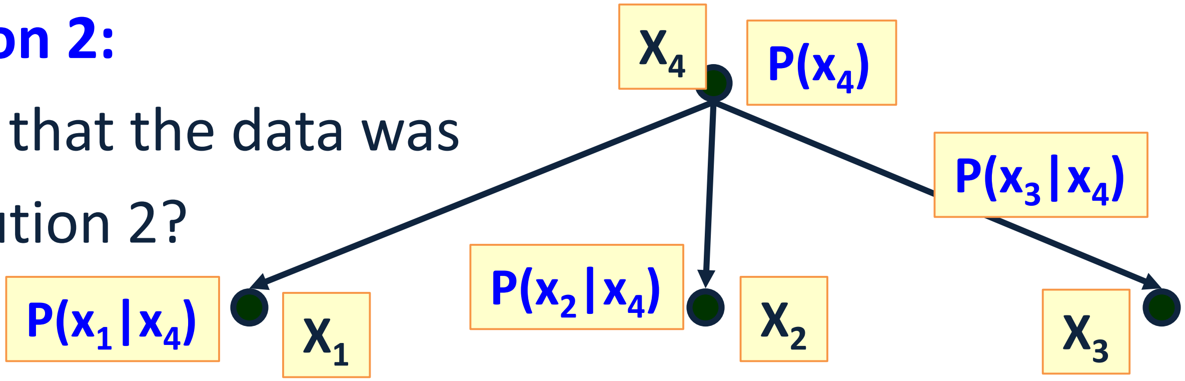
- What is the likelihood that this table generated the data?

$$P(T|D) = P(D|T)\, P(T)/P(D)$$

- Likelihood(T) $\sim= P(D|T) \sim= P(1011|T)\ P(1001|T)P(0100|T)$
  - $P(1011|T)= 0$
  - $P(1001|T)= 0.1$
  - $P(0100|T)= 0.1$
- $P(\text{Data}|\text{Table})=0$

# Example: Learning Distributions

- **Probability Distribution 2:**
- What is the likelihood that the data was sampled from Distribution 2?
- Need to define it:



$X_4$

$P(x_4)$

$P(x_3|x_4)$

$P(x_1|x_4)$

$X_1$

$P(x_2|x_4)$

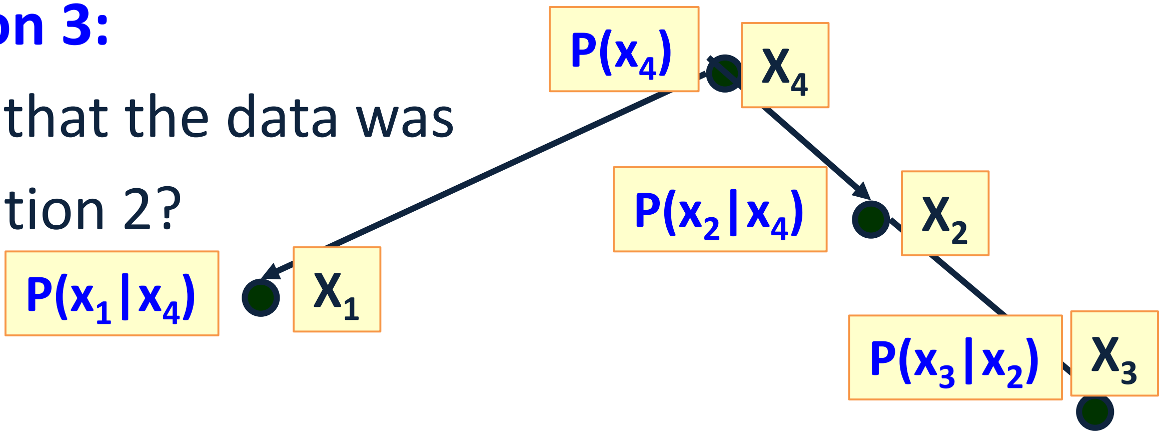
$X_2$

$X_3$

  - $P(x_4=1)=1/2$
  - $p(x_1=1|x_4=0)=1/2$ $p(x_1=1|x_4=1)=1/2$
  - $p(x_2=1|x_4=0)=1/3$ $p(x_2=1|x_4=1)=1/3$
  - $p(x_3=1|x_4=0)=1/6$ $p(x_3=1|x_4=1)=5/6$
- Likelihood(T) ~= P(D|T) ~= P(1011|T) P(1001|T)P(0100|T)
  - $P(1011|T)= p(x_4=1)p(x_1=1|x_4=1)p(x_2=0|x_4=1)p(x_3=1|x_4=1)=1/2\ 1/2\ 2/3\ 5/6= 10/72$
  - $P(1001|T)= \quad = 1/2\ 1/2\ 2/3\ 5/6=10/72$
  - $P(0100|T)= \quad =1/2\ 1/2\ 2/3\ 5/6=10/72$
  - $P(Data|Tree)=125/4*3^6$

# Example: Learning Distributions

- **Probability Distribution 3:**
- What is the likelihood that the data was sampled from Distribution 2?
- Need to define it:



  - $P(x_4=1)=2/3$
  - $p(x_1=1|x_4=0)=1/3$ $\quad$ $p(x_1=1|x_4=1)=1$
  - $p(x_2=1|x_4=0)=1$ $\quad$ $p(x_2=1|x_4=1)=1/2$
  - $p(x_3=1|x_2=0)=2/3$ $\quad$ $p(x_3=1|x_2=1)=1/6$
- Likelihood(T) ~= P(D|T) ~= P(1011|T) P(1001|T)P(0100|T)
  - P(1011|T)= $p(x_4=1)p(x_1=1|x_4=1)p(x_2=0|x_4=1)p(x_3=1|x_2=1)$=2/3 1 1/2 2/3= 2/9
  - P(1001|T)= = 1/2 1/2 2/3 1/6=1/36
  - P(0100|T)= =1/2 1/2 1/3 5/6=5/72
  - P(Data|Tree)=10/ $3^6 2^6$

Distribution 2 is the most likely distribution to have produced the data.

# Example: Summary

- We are now in the same situation we were when we decided which of two coins, fair (0.5,0.5) or biased (0.7,0.3) generated the data.
- But, this isn't the most interesting case.
- In general, we will not have a small number of possible distributions to choose from, but rather a parameterized family of distributions. (analogous to a coin with p Є [0,1] )

- We need a systematic way to search this family of distributions.

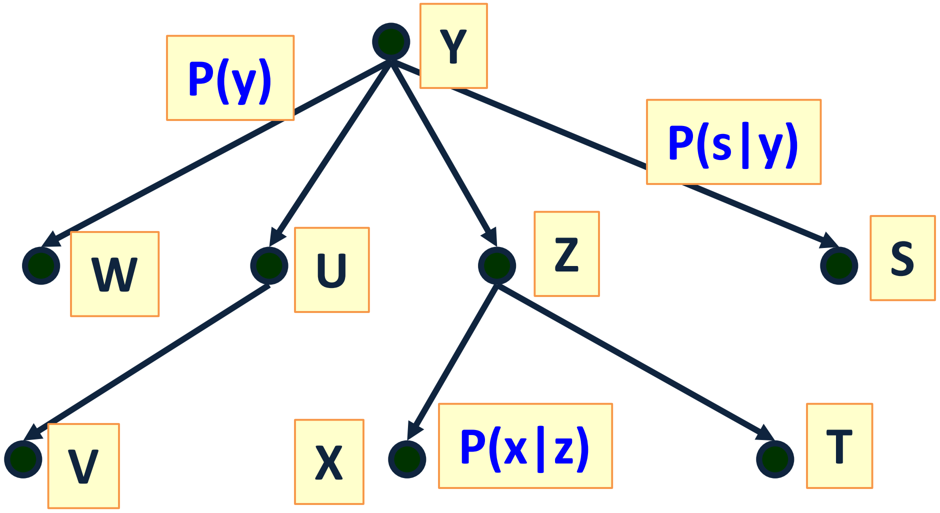
# Example: Summary

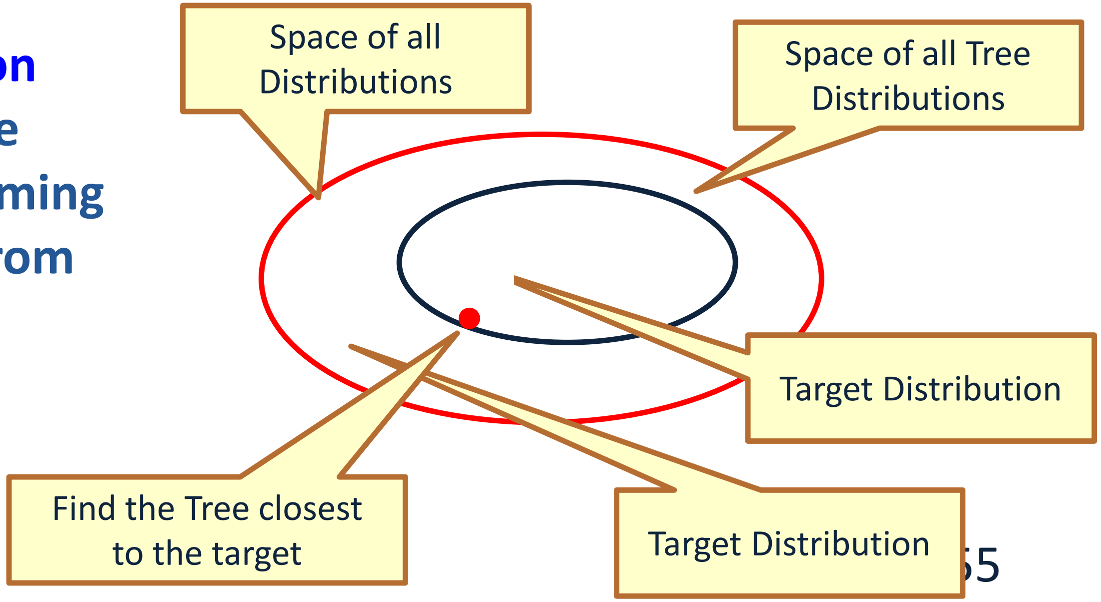
- First, let's make sure we understand what we are after.

- We have 3 data points that have been generated according to our target distribution:  1011; 1001; 0100
- What is the target distribution ?
  - We cannot find THE target distribution.
- What is our goal?
  - As before – we are interested in generalization –
  - Given Data (e.g., the above 3 data points), we would like to know P(1111) or P(11**), P(***0) etc.
- We could compute it directly from the data, but….
  - Assumptions about the distribution are crucial here

# Learning Tree Dependent Distributions

- **Learning Problem:**
  - **1. Given data (n tuples) assumed to be sampled from a tree-dependent distribution**
  - **find the most probable tree representation of the distribution.**
  - **2. Given data (n tuples)**
  - **find the tree representation that best approximates the distribution (without assuming that the data is sampled from a tree-dependent distribution.)**




Space of all Distributions

Space of all Tree Distributions

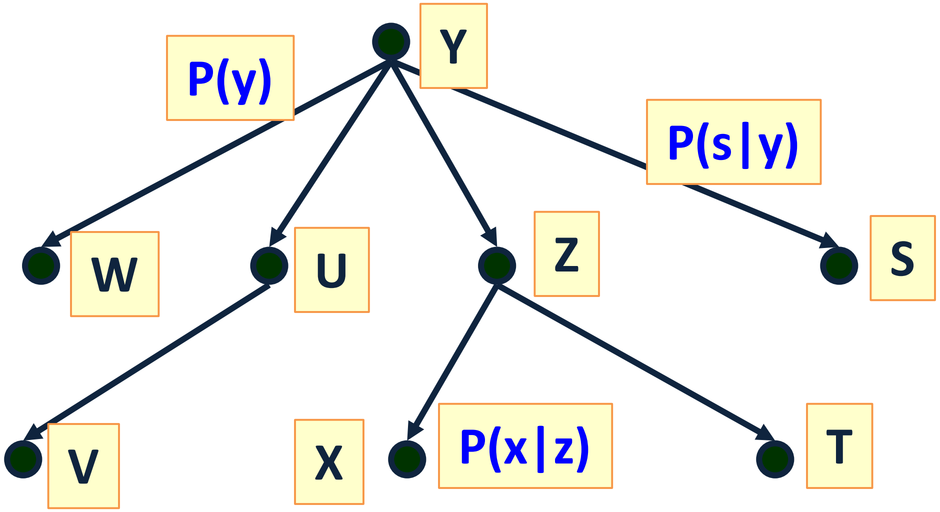
Target Distribution

Find the Tree closest to the target

Target Distribution

# Learning Tree Dependent Distributions

- **Learning Problem:**
  - **1. Given data (n tuples) assumed to be sampled from a tree-dependent distribution**
  - **find the most probable tree representation of the distribution.**
  - **2. Given data (n tuples)**
  - **find the tree representation that best approximates the distribution (without assuming that the data is sampled from a tree-dependent distribution.)**



- The simple minded algorithm for learning a tree dependent distribution requires
- (1) for each tree, compute its likelihood

$L(T) = P(D|T) =$

$$=\text{argmax}_T \prod_{\{x\}} P_T (x_1, x_2, \ldots x_n) =$$

$$=\text{argmax}_T \prod_{\{x\}} P_T (x_i|\text{Parents}(x_i))$$

(2) Find the maximal one

# 1. Distance Measure

- To measure how well a probability distribution P is approximated by probability distribution T we use here the Kullback-Leibler cross entropy measure (KL-divergence):

$$\mathbf{D(P,T)} = \sum_{x} \mathbf{P(x)\log\frac{P(x)}{T(x)}}$$

- Non negative.
- D(P,T)=0 iff P and T are identical
- Non symmetric. Measures how much P differs from T.

# 2. Ranking Dependencies

- Intuitively, the important edges to keep in the tree are edges (x---y) for x, y which depend on each other.
- Given that the distance between the distribution is measured using the KL divergence, the corresponding measure of dependence is the mutual information between x and y, (measuring the information x gives about y)

$$\mathbf{I(x,y) = \sum_{x,y} P(x,y)\log \frac{P(x,y)}{P(x)P(y)}}$$

- which we can estimate with respect to the empirical distribution (that is, the given data).

# Learning Tree Dependent Distributions

- The algorithm is given m independent measurements  from P.
- For each variable x, estimate P(x)   (Binary variables – n numbers)
- For each pair of variables x, y, estimate P(x,y) (O(n$^2$) numbers)
- For each pair of variables compute  the mutual information
- Build a complete undirected graph with all the variables as vertices.
- Let I(x,y) be the weights of the edge (x,y)
- Build a maximum weighted spanning tree

# Spanning Tree

- Goal: Find a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is maximized
- Sort the weights
- Start greedily with the largest one.
- Add the next largest as long as it does not create a loop.
- In case of a loop, discard this weight and move on to the next weight.
- This algorithm will create a tree;
- It is a spanning tree: it touches all the vertices.
- It is not hard to see that this is the maximum weighted spanning tree
- The complexity is $O(n^2 \log(n))$

# Learning Tree Dependent Distributions

- The algorithm is given m independent measurements from P.
- For each variable x, estimate P(x) (Binary variables – n numbers)
- For each pair of variables x, y, estimate P(x,y) (O($n^2$) numbers)
- For each pair of variables compute the mutual information
- Build a complete undirected graph with all the variables as vertices.
- (2) Let I(x,y) be the weights of the edge (x,y)
- Build a maximum weighted spanning tree
- (3) Transform the resulting undirected tree to a directed tree.
  - Choose a root variable and set the direction of all the edges away from it.
- (1) Place the corresponding conditional probabilities on the edges.

# Correctness (1)

- Place the corresponding conditional probabilities on the edges.
- Given a tree t, defining probability distribution T by forcing the conditional probabilities along the edges to coincide with those computed from a sample taken from P, gives the best tree dependent approximation to P
- Let T be the tree-dependent distribution according to the fixed tree t.

$$\mathbf{T(x) = \Pi\ T(x_i | Parent(x_i)) = \Pi\ P(x_i | \pi\ (x_i))}$$

- Recall:

$$\mathbf{D(P,T) = \sum_{x} P(x) \log \frac{P(x)}{T(x)}}$$

# Correctness (1)

- Place the corresponding conditional probabilities on the edges.
- Given a tree t, defining T by forcing the conditional probabilities along the edges to coincide with those computed from a sample taken from P, gives the best t-dependent approximation to P

$$\mathbf{D(P,T)} = \sum_{\mathbf{x}} \mathbf{P(x)\log\frac{P(x)}{T(x)}} =$$

$$= \sum_{\mathbf{x}} \mathbf{P(x)\log P(x)} - \sum_{\mathbf{x}} \mathbf{P(x)\log T(x)} =$$

$$= -\mathbf{H(x)} - \sum_{\mathbf{x}} \mathbf{P(x)} \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{\log T(x_i \mid \pi(x_i))} =$$

Slight abuse of notation at the root

- When is this maximized?
  - That is, how to define $T(x_i|\pi(x_i))$?

# Correctness (1)

$$D(P,T) = \sum_{x} P(x)\log\frac{P(x)}{T(x)} = \sum_{x} P(x)\log P(x) - \sum_{x} P(x)\log T(x) =$$

$$= -H(x) - \sum_{x} P(x)\sum_{i=1}^{n} \log T(x_i \mid \pi(x_i)) =$$

$$= -H(x) - E_P[\sum_{i=1}^{n} \log T(x_i \mid \pi(x_i))] =$$

$$= -H(x) - \sum_{i=1}^{n} E_P[\log T(x_i \mid \pi(x_i))] =$$

Definition of expectation:

$$= -H(x) - \sum_{i=1}^{n} \sum_{(x_{i,},\pi(x_i))} P(x_i,\pi(x_i)) \log T(x_i \mid \pi(x_i)) =$$

$$= -H(x) - \sum_{i=1}^{n} \sum_{\pi(x_i)} P(\pi(x_i)) \underline{\sum_{x_i} P(x_i \mid \pi(x_i)) \log T(x_i \mid \pi(x_i))}$$

$\sum_i P(x_i|(x_i) \log T(x_i|(x_i))$ takes its maximal value when we set: $T(x_i|(x_i)) = P(x_i|(x_i))$

# Correctness (2)

- Let I(x,y) be the weights of the edge (x,y). Maximizing the sum of the information gains minimizes the distributional distance.
- We showed that: $\mathbf{D(P,T) = -H(x) - \sum_{i=1}^{n} \sum_{\underline{(x_{i,},\pi(x_i))}} P(x_i,\pi(x_i)) \log P(x_i \mid \pi(x_i))}$
- However: $$\mathbf{\log P(x_i \mid \pi(x_i)) = \log \frac{P(x_i,\pi(x_i))}{P(x_i)P(\pi(x_i))} + \log P(x_i)}$$

$$\mathbf{P(x_i,\pi(x_i))\log P(x_i \mid \pi(x_i)) = \underline{P(x_i,\pi(x_i))\log \frac{P(x_i,\pi(x_i))}{P(x_i)P(\pi(x_i))}} + P(x_i,\pi(x_i))\log P(x_i)}$$

- This gives:

$$\mathbf{D(P,T) = -H(x) - \sum_{1,n} \underline{I(x_i,(x_i))} - \sum_{1,n}\underline{\sum_x P(x_i)} \log P(x_i)}$$

- 1st and 3rd term do not depend on the tree structure. Since the distance is non negative, minimizing it is equivalent to maximizing the sum of the edges weights I(x,y) .

# Correctness (2)

- Let I(x,y) be the weights of the edge (x,y). Maximizing the sum of the information gains minimizes the distributional distance.
- We showed that the T is the best tree approximation of P if it is chosen to maximize the sum of the edges weights.

$$D(P,T) = -H(x) - \sum_{1,n} I(x_i,(x_i)) - \sum_{1,n}\sum_{x} P(x_i) \log P(x_i)$$

- The minimization problem is solved without the need to exhaustively consider all possible trees.
- This was achieved since we transformed the problem of finding the best tree to that of finding the heaviest one, with mutual information on the edges.

# Correctness (3)

- Transform the resulting undirected tree to a directed tree.
  (Choose a root variable and direct of all the edges away from it.)
  - What does it mean that you get the same distribution regardless of the chosen root? (Exercise)
- This algorithm learns the best tree-dependent approximation of a distribution D.

$$L(T) = P(D|T) = \prod_i P_T(x_i|\mathrm{Parent}(x_i))$$

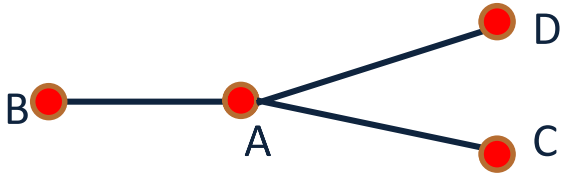
- Given data, this algorithm finds the tree that maximizes the likelihood of the data.
- The algorithm is called the Chow-Liu Algorithm. Suggested in 1968 in the context of data compression, and adapted by Pearl to Bayesian Networks. Invented a couple more times, and generalized since then.

# Example: Learning tree Dependent Distributions

- We have 3 data points that have been generated according to the target distribution: 1011; 1001; 0100

$$\mathbf{I(x,y)} = \sum_{x,y} \mathbf{P(x,y)\log \frac{P(x,y)}{P(x)P(y)}}$$

- We need to estimate some parameters:
- P(A=1) = 2/3, P(B=1)=1/3, P(C=1)=1/3), P(D=1)=2/3
- For the values 00, 01, 10, 11 respectively, we have that:
- P(A,B)=0; 1/3; 2/3; 0 P(A,B)/P(A)P(B)=0; 3; 3/2; 0 I(A,B) ~ 9/2
- P(A,C)=1/3; 0; 1/3; 1/3 P(A,C)/P(A)P(C)=3/2; 0; 3/4; 3/2 I(A,C) ~ 15/4
- P(A,D)=1/3; 0; 0; 2/3 P(A,D)/P(A)P(D)=3; 0; 0; 3/2 I(A,D) ~ 9/2
- P(B,C)=1/3; 1/3; 1/3;0 P(B,C)/P(B)P(C)=3/4; 3/2; 3/2; 0 I(B,C) ~ 15/4
- P(B,D)=0; 2/3; 1/3;0 P(B,D)/P(B)P(D)=0; 3; 3/2; 0 I(B,D) ~ 9/2
- P(C,D)=1/3; 1/3; 0; 1/3 P(C,D)/P(C)P(D)=3/2; 3/4; 0; 3/2 I(C,D) ~ 15/4
- Generate the tree; place probabilities.



# Learning tree Dependent Distributions

- Chow-Liu algorithm finds the tree that maximizes the likelihood.
- In particular, if D is a tree dependent distribution, this algorithm learns D. (what does it mean ?)
- Less is known on how many examples are needed in order for it to converge. (what does that mean?)
- Notice that we are taking statistics to estimate the probabilities of some event in order to generate the tree. Then, we intend to use it to evaluate the probability of other events.
- One may ask the question: why do we need this structure ? Why can't answer the query directly from the data ?
- (Almost like making prediction directly from the data in the badges problem)