



Bayesian Networks

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), or by other authors who have made their ML slides available.

Administration (12/07/20)

Are we recording? YES!

Available on the web site

- Remember that all the lectures are available on the website **before the class**
 - **Go over it and be prepared**
 - **A new set** of written notes will accompany most lectures, with some more details, examples and, (when relevant) some code.
- **HW4** is due today.
- **HW5** is out; due 12/10 (last day of the semester).
 - It is mostly a summary of the material we covered this semester (with a focus on the second half) and will help you prepare for the exam.
 - No programming.
 - We will give an extension until a few days before the exam.
- We meet **three** time (Monday, Wednesday, **Thursday**) this week.
- The **Final** is on 12/18.
 - Similar style to the mid-term. Comprehensive, with emphasis on the material after the mid-term.
 - 90 minutes. You can start it any time between 9am and 7:30 pm ET. Your 90 minutes will be measure from the time you start.
 - Between 10:30-noon ET, most of the TAs will be on-board to respond to clarification questions. If possible, do it at that time.
 - There will be at least one TA available throughout the day to respond to clarification questions.
 - Communication will only be done via private Piazza posts.

Projects

- CIS 519 students need to do a team project: Read the [project descriptions](#) and follow the updates on the [Project webpage](#)
 - Teams will be of size 2-4
 - We will help grouping if needed
- There will be 3 options for projects.
 - Natural Language Processing (Text)
 - Computer Vision (Images)
 - Speech (Audio)
- In all cases, we will give you datasets and initial ideas
 - The problem will be multiclass classification problems
 - You will get annotated data only for some of the labels, but will also have to predict other labels
 - 0-zero shot learning; few-shot learning; transfer learning
- A detailed note will come out today.
- Timeline:
 - 11/11 Choose a project and team up
 - 11/23 Initial proposal describing what your team plans to do
 - 12/2 Progress report: 1 page. What you have done; plans; problems.
 - 12/21 Final paper + short video
- Try to make it interesting!

So far...

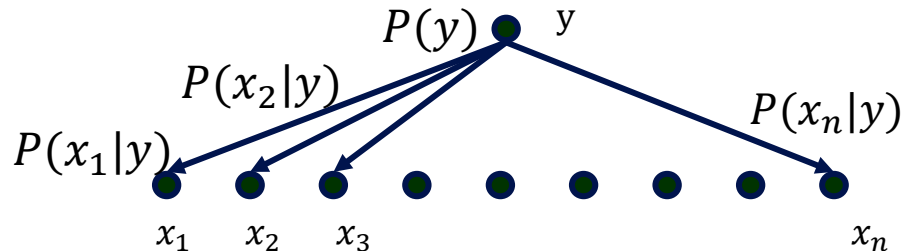
- Bayesian Learning
 - What does it mean to be Bayesian?
- Naïve Bayes
 - Independence assumptions
- EM Algorithm
 - Learning with hidden variables
- Today:
 - Representing arbitrary probability distributions
 - Inference
 - Exact inference; Approximate inference
 - Learning Representations of Probability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

Unsupervised Learning

- We get as input $(n + 1)$ tuples: $(\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n, \mathbf{X}_{n+1})$
- 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.

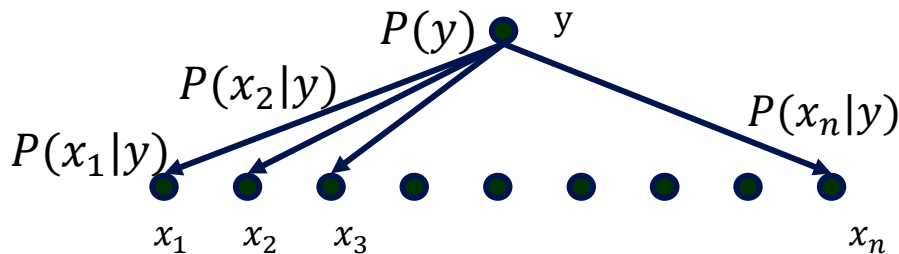
Simple Distributions

- 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)
 - $P(x_i | x_j, y) = P(x_i|y) \forall i, j$
- 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 $P(x_i | x_j, y) = P(x_i | y) \forall i, j$ we can compute the joint probability distribution on the $n + 1$ variables
 - $P(y, x_1, x_2, \dots, x_n) = p(y) \prod_1^n P(x_i | y)$
- Therefore, we can compute the probability of any event:
 - $P(x_1 = 0, x_2 = 0, y = 1) =$
- More efficiently (directly from the independence assumption):
 - $P(x_1 = 0, x_2 = 0, y = 1) = P(x_1 = 0, x_2 = 0 | y = 1) p(y = 1) =$
 - $= P(x_1 = 0 | y = 1) P(x_2 = 0 | y = 1) p(y = 1)$
- We can compute the probability of any event or conditional event over the $n + 1$ variables.

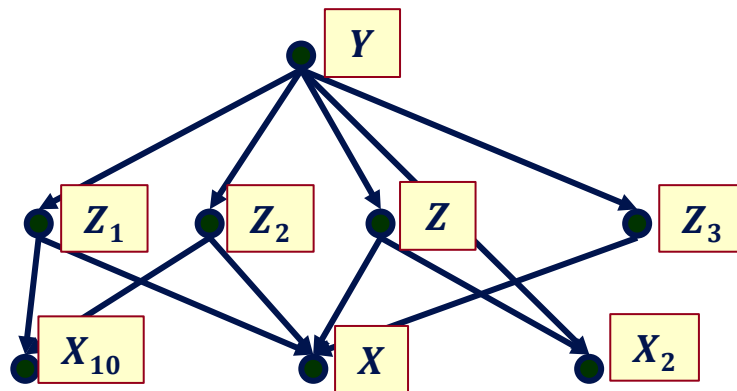
Representing Probability Distribution

- Goal: To represent all joint probability distributions over a set of random variables X_1, X_2, \dots, 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: $\forall x, x$ is independent of its non-descendants given its parents

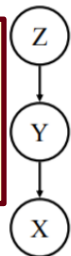
This is a theorem. To prove it, order the nodes from leaves up, and use the product rule. The terms are called CPTs (Conditional Probability tables) and they completely define the probability distribution.



z is a parent of x

x is a descendant of y

- Show that: $P(X|Y,Z) = P(X|Y)$.
- BTW, since independence is symmetric, you can also show that $P(Z|Y,X) = P(Z|Y)$



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

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

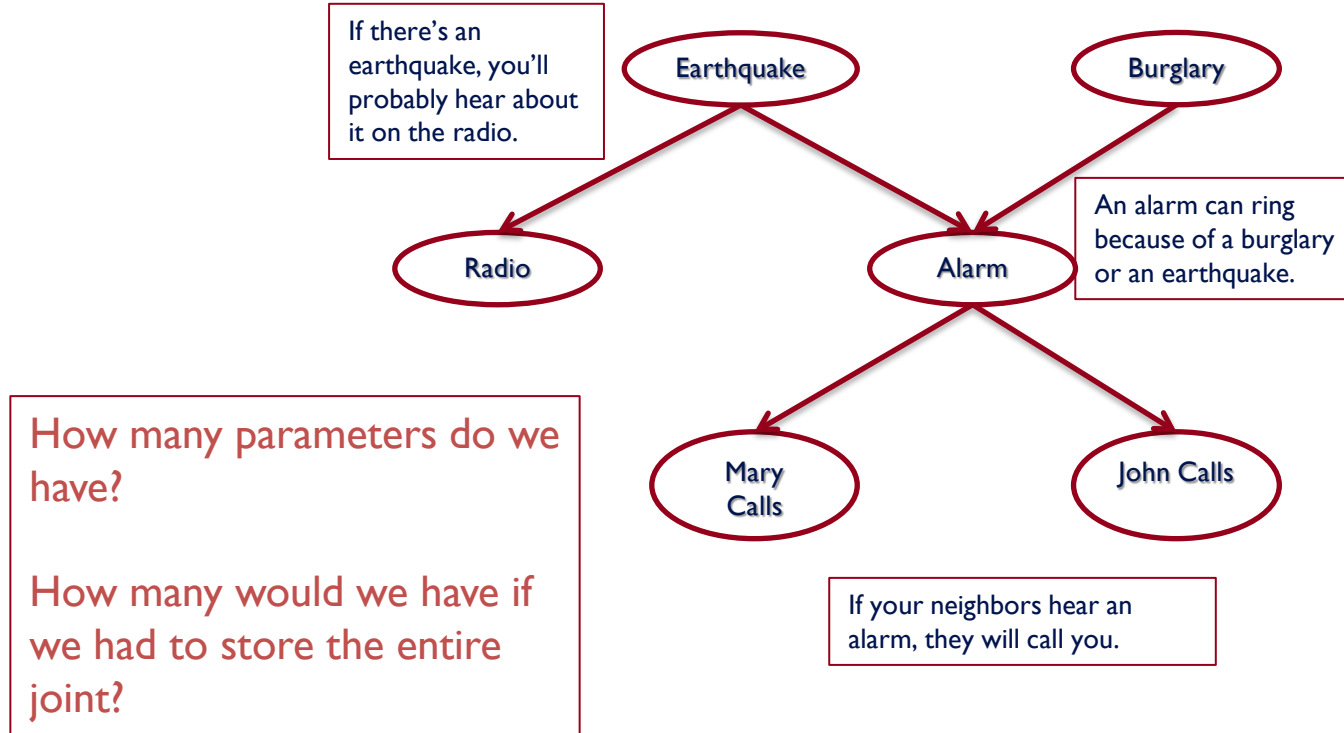
Bayesian Network

- 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?

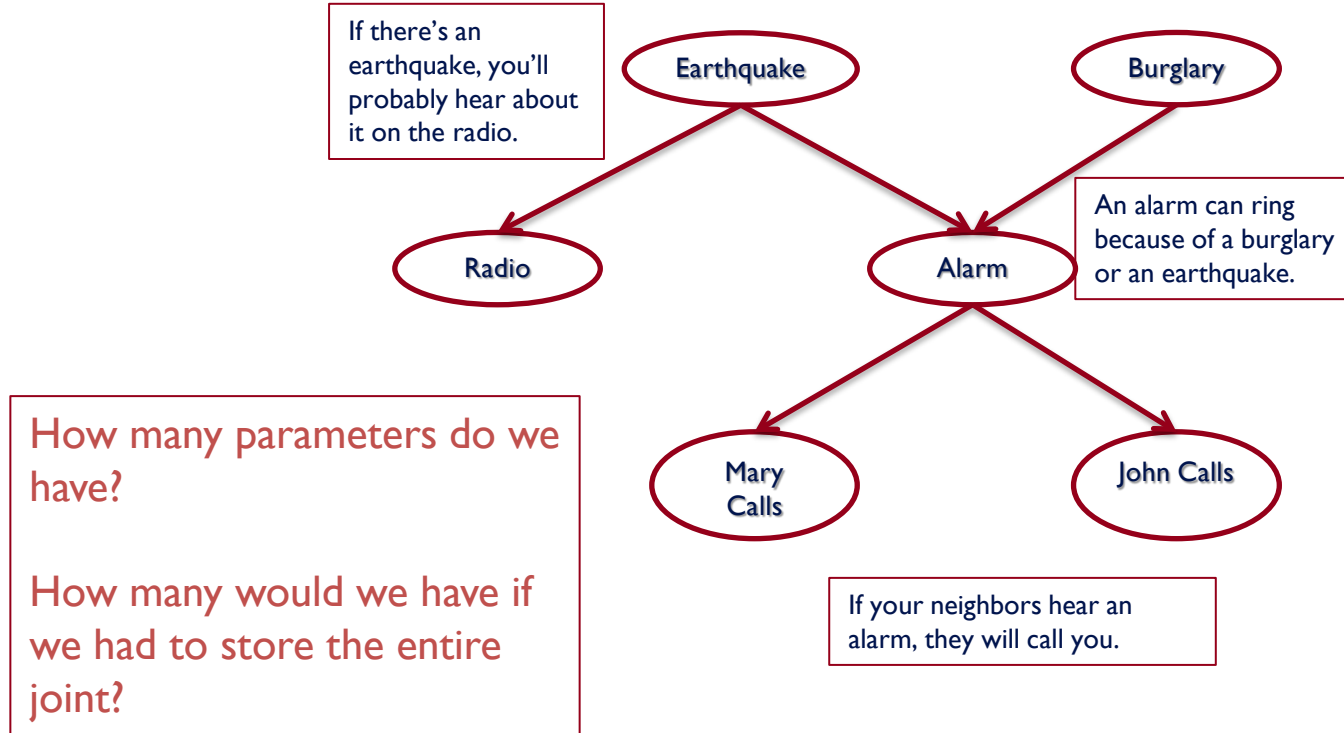
Bayesian Network: Example







How many independent parameters are needed to represent a probability distribution over these 6 variables?

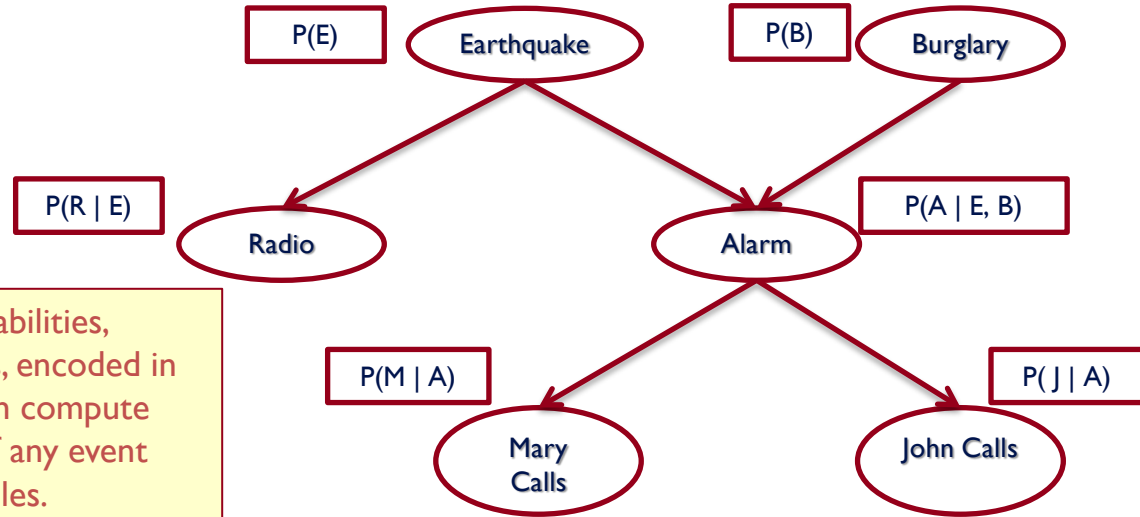
Bayesian Network: Example





How many independent parameters are needed to represent $[P(E), P(R|E), P(A|E,B)]$?

Bayesian Network: Example



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

$$\begin{aligned} P(E, B, A, R, M, J) &= P(E) P(B, A, R, M, J | E) = \\ &= P(E) P(B) P(A, R, M, J | E, B) = \\ &= P(E) P(B) P(R | E, B) P(M, J, A | E, B) \\ &= P(E) P(B) P(R | E) P(M, J | A, E, B) P(A | E, B) \\ &= P(E) P(B) P(R | E) P(M | A) P(J | A) P(A | E, B) \end{aligned}$$

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 (evidence) E , what is the probability of assignment Y ?
 - $P(R = \text{off}, A = \text{off} \mid E = e) = ?$ (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 (evidence) E , what is the most likely assignment to Y ?
 - $\text{Argmax}_y P(Y = y \mid E = e)$ (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

Administration (12/09/20)

Are we recording? YES!

Available on the web site

- Remember that all the lectures are available on the website **before the class**
 - **Go over it and be prepared**
 - **A new set of written notes** will accompany most lectures, with some more details, examples and, (when relevant) some code.
- **HW5** is out; due 12/10 (last day of the semester).
 - It is mostly a summary of the material we covered this semester (with a focus on the second half) and will help you prepare for the exam. No programming.
 - **An extension until Tuesday 12/15 – no additional slack day. This is a hard deadline.**
- We meet **three** time (Monday, Wednesday, **Thursday**) this week.
- The **Final** is on 12/18.
 - Similar style to the mid-term. Comprehensive, with emphasis on the material after the mid-term.
 - 90 minutes. You can start it any time between 9am and 7:30 pm ET. Your 90 minutes will be measure from the time you start.
 - Between 10:30-noon ET, most of the TAs will be on-board to respond to clarification questions. If possible, do it at that time.
 - There will be at least one TA available throughout the day to respond to clarification questions.
 - Communication will only be done via private Piazza posts.

Projects

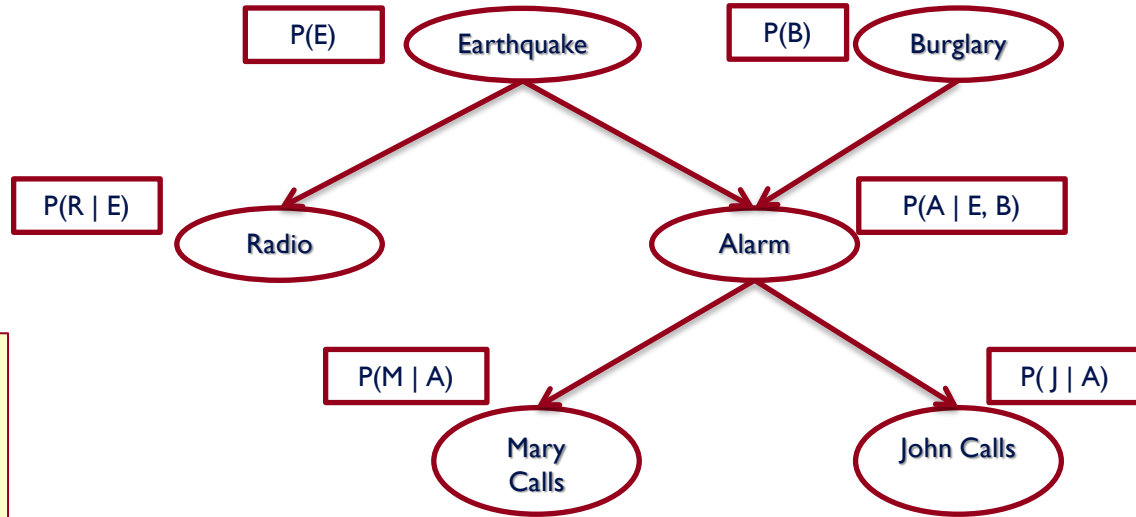
- CIS 519 students need to do a team project: Read the [project descriptions](#) and follow the updates on the [Project webpage](#)
 - Teams will be of size 2-4
 - We will help grouping if needed
- There will be 3 options for projects.
 - Natural Language Processing (Text)
 - Computer Vision (Images)
 - Speech (Audio)
- In all cases, we will give you datasets and initial ideas
 - The problem will be multiclass classification problems
 - You will get annotated data only for some of the labels, but will also have to predict other labels
 - 0-zero shot learning; few-shot learning; transfer learning
- A detailed note will come out today.
- Timeline:
 - 11/11 Choose a project and team up
 - 11/23 Initial proposal describing what your team plans to do
 - 12/2 Progress report: 1 page. What you have done; plans; problems.
 - 12/21 Final paper + short video
- Try to make it interesting!

Bayesian Network: Example

How many parameters do we have?

How many would we have if we had to store the entire joint on 6 variables, without any independence assumptions?

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



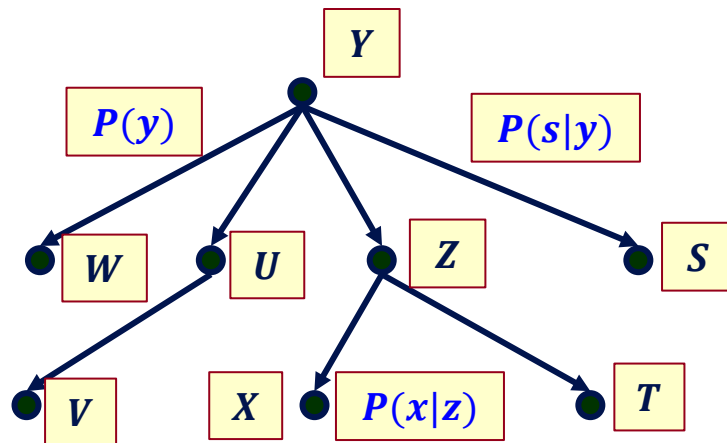
$$\begin{aligned} P(E, B, A, R, M, J) &= P(E) P(B, A, R, M, J | E) = \\ &= P(E) P(B) P(A, R, M, J | E, B) = \\ &= P(E) P(B) P(R | E, B) P(M, J, A | E, B) \\ &= P(E) P(B) P(R | E) P(M, J | A, E, B) P(A | E, B) \\ &= P(E) P(B) P(R | E) P(M | A) P(J | A) P(A | E, B) \end{aligned}$$

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 given z ; v is independent of w given u ;)

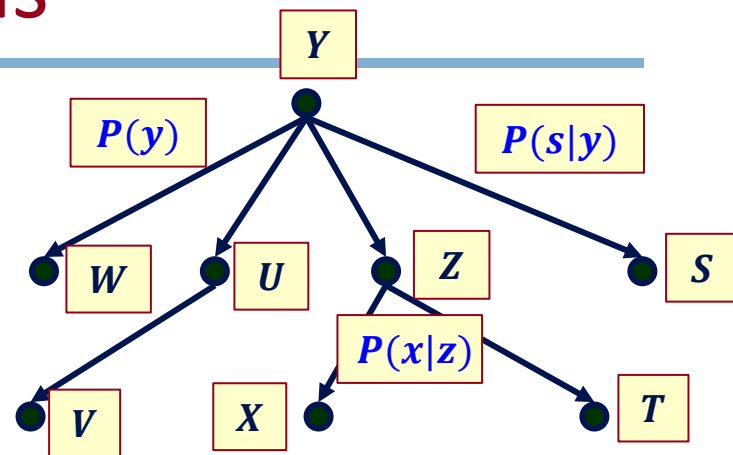
$$P(y, x_1, x_2, \dots, x_n) = p(y) \prod_i P(x_i | \text{Parents}(x_i))$$

- 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 $p(x)$?



- $P(x = 1) =$
 $= P(x = 1|z = 1)P(z = 1) + P(x = 1|z = 0)P(z = 0)$
- Recursively, go up the tree:

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

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

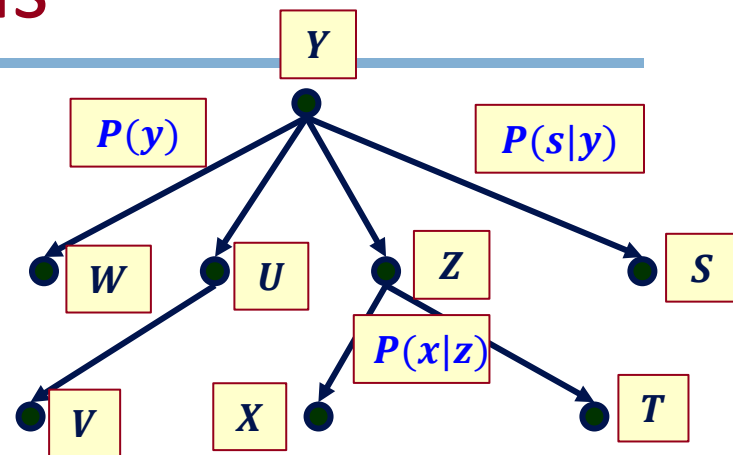
- Linear Time Algorithm

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

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

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 $p(x, y)$?



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

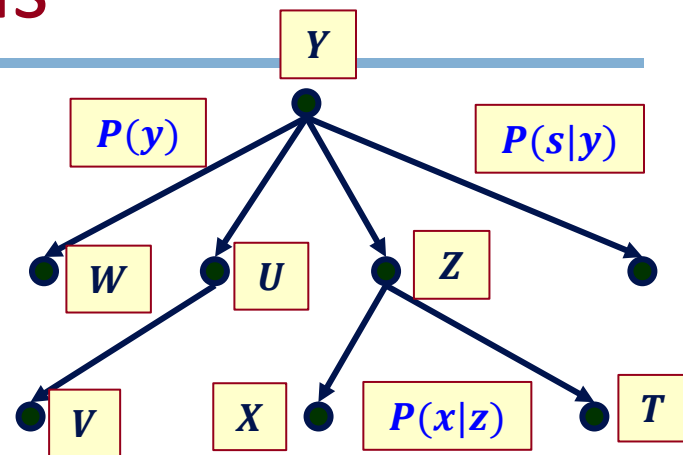
Tree Dependent Distributions

- 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)

$$P(x = 1, u = 0) = P(x = 1|u = 0)P(u = 0)$$

- Let y be a parent of x and u (we always have one)

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



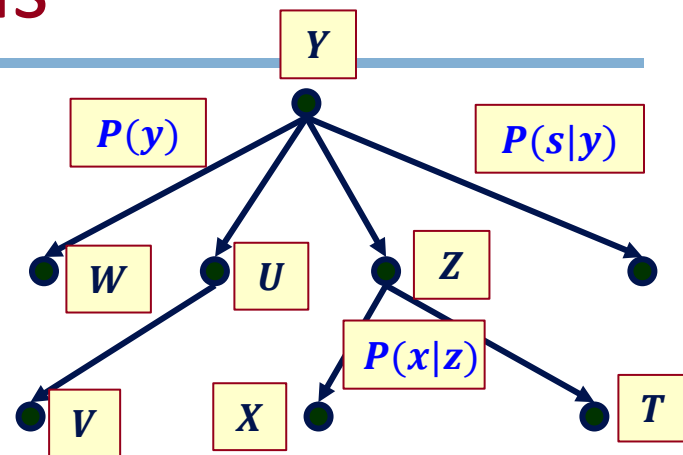
Now we have reduced it to cases we have seen

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

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 in this case is a simple application of Bayes rule and probability theory.



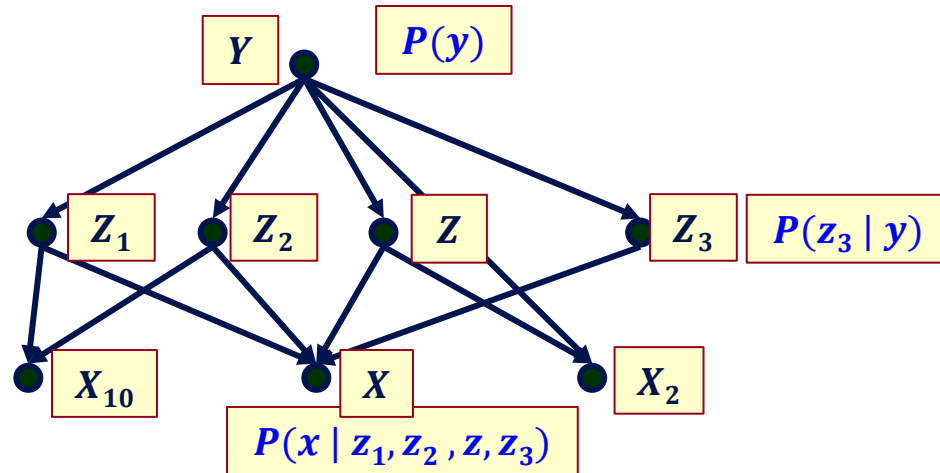
$$P(y, x_1, x_2, \dots, x_n) = p(y) \prod_i P(x_i | \text{Parents}(x_i))$$

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)



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

Variable Elimination

$$P(x_1, x_2, \dots, x_n) = \prod_i P(x_i | \text{Parents}(x_i))$$

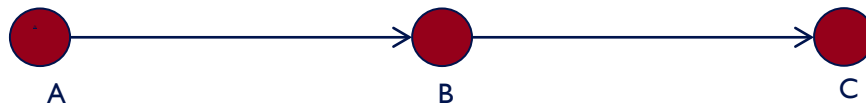
- Suppose the query is $P(X_1)$

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

$$P(x_1) = \sum_{x_2} \sum_{x_3} \dots \sum_{x_n} \underline{\prod_i P(x_i | \text{Parents}(x_i))}$$

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

Variable Elimination: Example 1



- We want to compute $P(C)$

$$\begin{aligned} P(C) &= \sum_A \sum_B P(A, B, C) = \sum_A \sum_B P(A)P(B|A)P(C|B) \\ &= \sum_B P(C|B) \sum_A P(A)P(B|A) \longrightarrow \text{Let's call this } f_A(B) \\ &= \sum_B P(C|B)f_A(B) \end{aligned}$$

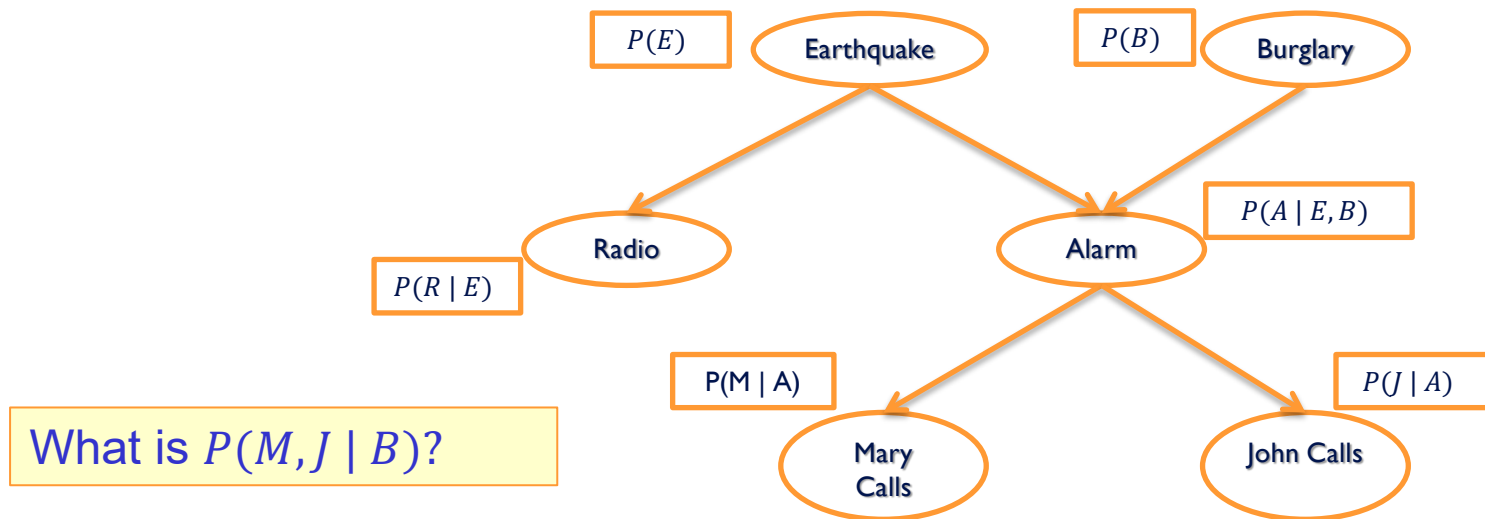
A has been (instantiated and) eliminated

- 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



$$\begin{aligned} P(E, B, A, R, M, J) &= \\ &= P(E) \cdot P(B) \cdot P(R|E) \cdot P(A|E, B) \cdot P(M|A) \cdot P(J|A) \end{aligned}$$

Variable Elimination: Example 2

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

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

It is sufficient to compute the numerator and normalize

Assumptions (graph;
joint representation)

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

Elimination order R, A, E

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

To eliminate R

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

Variable Elimination: Example 2

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

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

It is sufficient to compute the numerator and normalize

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

Elimination order A, E

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

To eliminate A

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

$$f_A(E, M, J) = \sum_A P(A|E, B = \text{true}) \cdot P(M|A) \cdot P(J|A)$$

Variable Elimination: Example 2

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

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

It is sufficient to compute the numerator and normalize

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

Finally eliminate E

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

Factors

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

$$f_A(E, M, J) = \sum_A P(A|E, B = \text{true}) \cdot P(M|A) \cdot P(J|A)$$

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

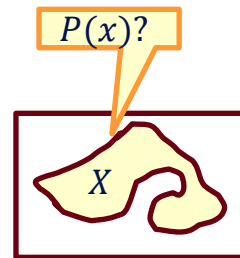
Inference

- 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.

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

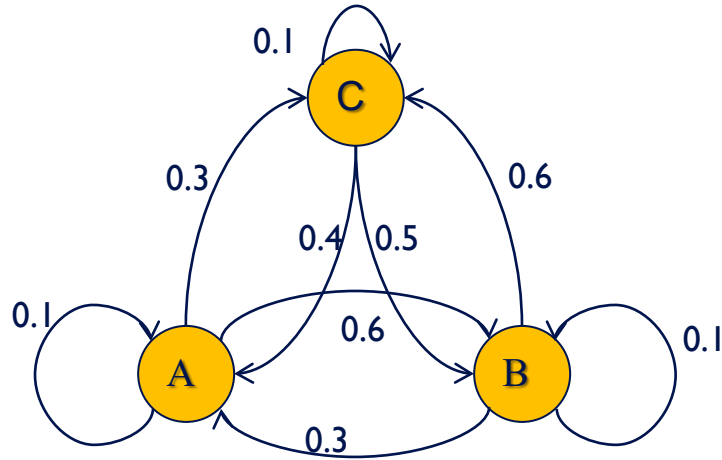


- For inference, we generate instances from the joint and count
- How do we generate instances?

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

$$P(X_0) \text{ and } P(X_{t+1} = i \mid X_t = j)$$

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

$$P(X = x_1 x_2 \dots x_n \mid e)$$

- 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

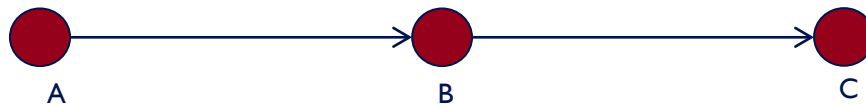
Gibbs Sampling

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

Gibbs Sampling: Big picture

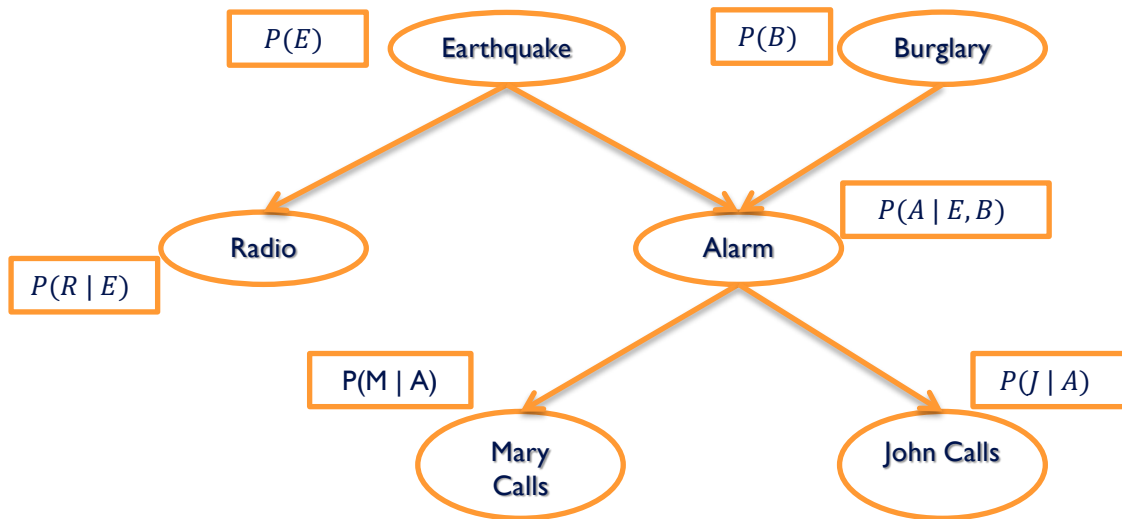
- Given some conditional distribution we wish to compute, collect samples from the Markov Chain
- Typically, the chain is allowed to run for some time before collecting samples (**burn in period**)
 - So that the chain settles into the stationary distribution
- Using the samples, we approximate the posterior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

Gibbs Sampling Example 1



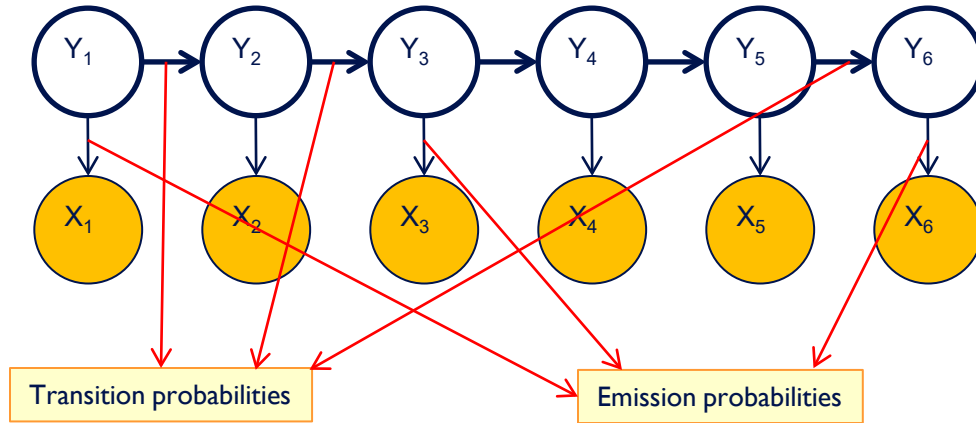
- We want to compute $P(C)$:
 - Suppose, after burn in, the Markov Chain is at $A = true, B = false, C = false$
1. Pick a *variable* $\rightarrow B$
 2. Draw the new value of B from
 - $P(B | A = true, C = false) = P(B | A = true)$
 - Suppose $B^{new} = true$
 3. Our new sample is $A = true, B = true, C = false$
 4. Repeat

Gibbs Sampling Example 2



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

Example: Hidden Markov Model



- A Bayesian Network with a specific structure.
- X 's are called the observations and Y 's 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
 - $P(X | \text{parameters})$: Dynamic Programming
- Finding the best hidden states for a given sequence
 - $P(Y | X, \text{parameters})$: Dynamic Programming
- Learning the parameters from observations
 - EM

Gibbs Sampling for HMM

- Goal: Computing $P(y|x)$
- Initialize the Y 's randomly
- Iterate:
 - Pick a random Y_i
 - Draw Y_i^t from $P(Y_i|Y_{i-1}Y_{i+1}, X_i)$
- Compute the probability using counts after the burn in period

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

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

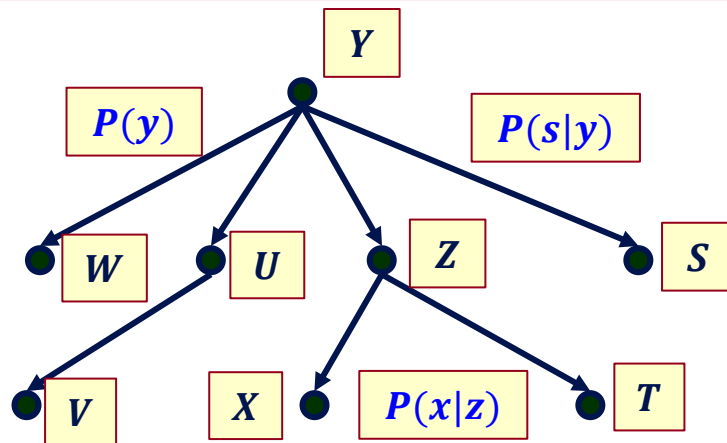
Bayesian Networks

- 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?
 - Think about it as a Generative model
 - Find the tree representation of the distribution.
 - What does that mean?
- Among all trees, find the **most likely** one, given the data:
$$P(\mathbf{T}|\mathbf{D}) = P(\mathbf{D}|\mathbf{T}) P(\mathbf{T})/P(\mathbf{D})$$

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



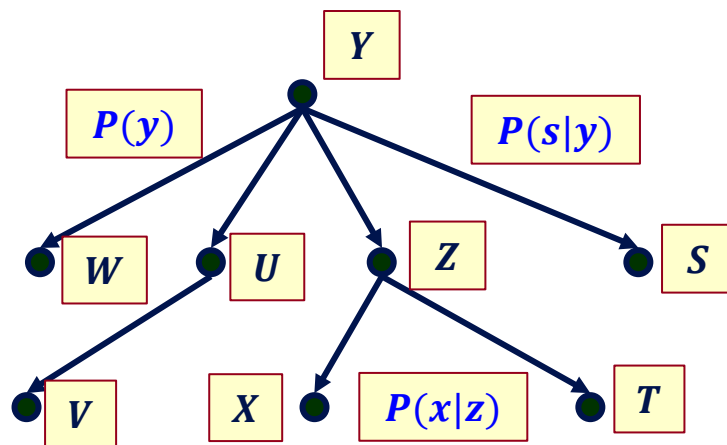
Tree Dependent Distributions

- 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 $P(D|T)$,

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

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



Tree Dependent Distributions

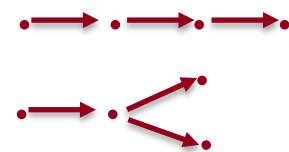
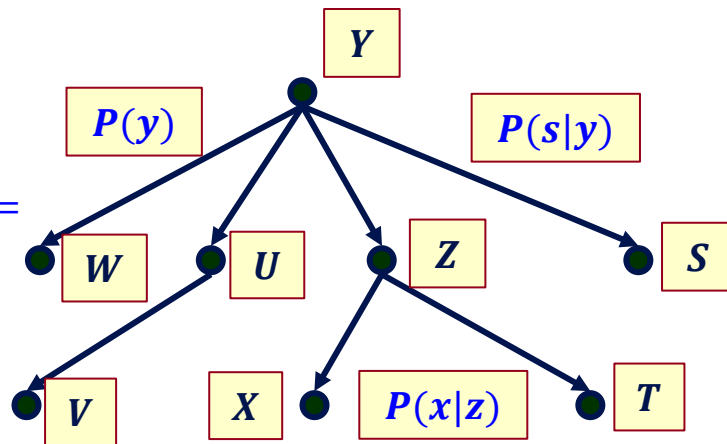
- Learning Problem:
 - Given data (n tuples) assumed to be sampled from a tree-dependent distribution
 - Find the tree representation of the distribution.
- Assuming uniform prior on trees, the Maximum Likelihood approach is to maximize $P(D|T)$,

$$T_{ML} = \operatorname{argmax}_T P(D|T) = \operatorname{argmax}_T \prod_{\{x\}} P_T(x_1, x_2, \dots, x_n) = \\ = \operatorname{argmax}_T \prod_{\{x\}} P_T(x_i | \operatorname{Parents}(x_i))$$

Try this for naïve Bayes

Next we will look at some examples over 4 Boolean variables

- Note that when we talk about trees, they could have multiple shapes.



How many independent parameters are needed to represent a distribution over 4 Boolean variable?

4

3

16

15

None of the above

How many independent parameters are needed to represent a tree dependent distribution over 4 Boolean variables?

It depends on the shape of the tree

None of the above

4

15

7

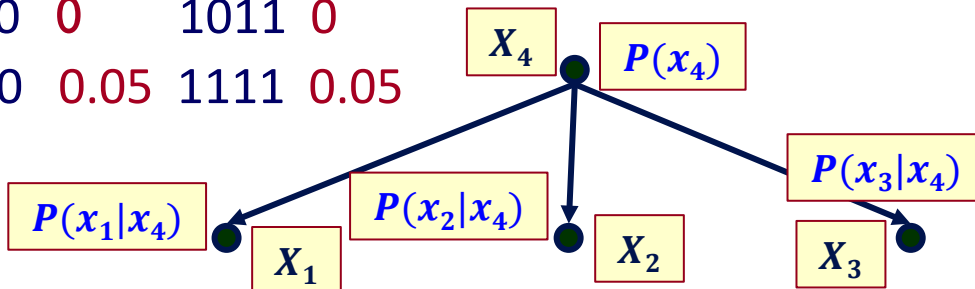
Example: Learning Distributions

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

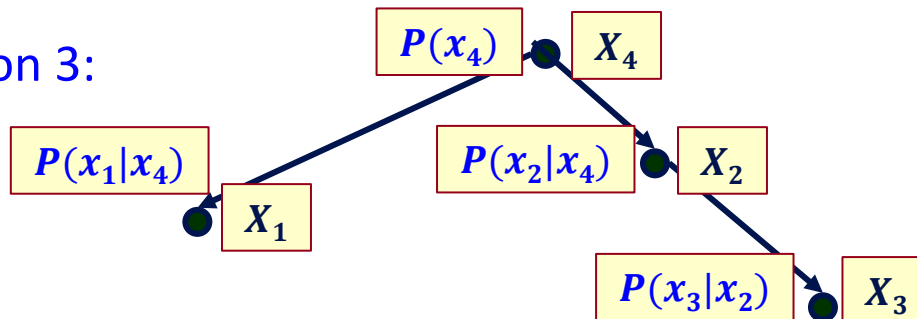
- Probability Distribution 1:

| | | | | | | | |
|------|------|------|------|------|------|------|------|
| 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:

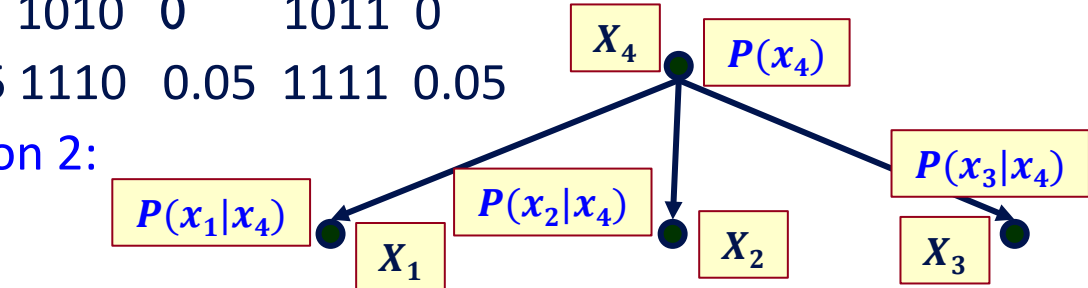


Example: Learning Distributions

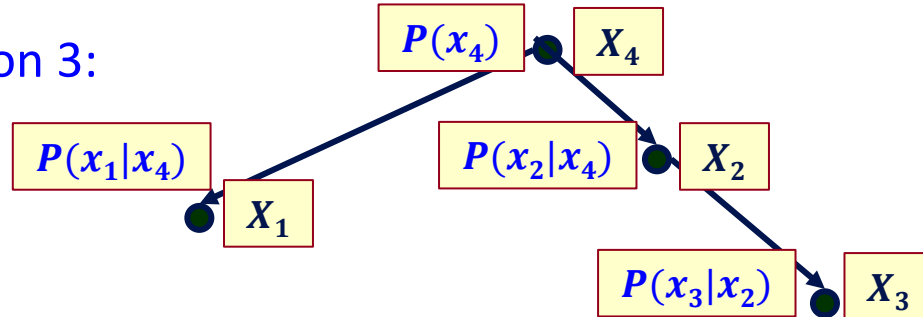
- Probability Distribution 1:

| | | | | | | | |
|------|------|------|------|------|------|------|------|
| 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:



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

Example: Learning Distributions

- 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?

We are given 3 data

points:

1011; 1001; 0100

Which one is the target
distribution?



What is the likelihood that the given dataset was sampled from Distribution 1?

Example: Learning Distributions

- 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?

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

- $Likelihood(T) \sim P(D|T) \sim P(1011|T) P(1001|T) P(0100|T)$

- $P(1011|T) = 0$

- $P(1001|T) = 0.1$

- $P(0100|T) = 0.1$

- $P(Data|Table) = 0$

We are given 3 data

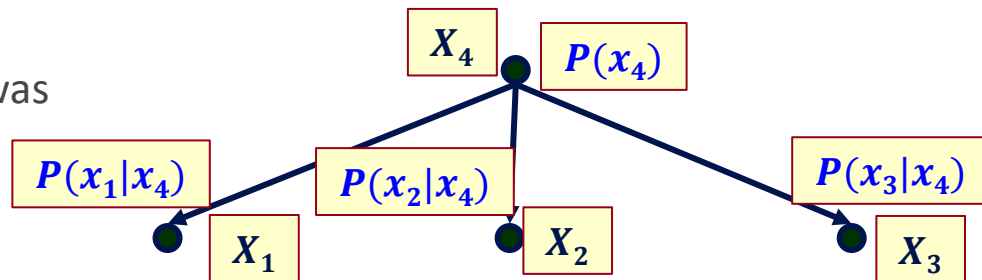
points:

1011; 1001; 0100

Which one is the target
distribution?

Example: Learning Distributions

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



- $P(x_4 = 1) = 1/2$
- $p(x_1 = 1|x_4 = 0) = 1/2$ $p(x_1 = 1|x_4 = 1) = 1/2$
- $p(x_2 = 1|x_4 = 0) = 1/3$ $p(x_2 = 1|x_4 = 1) = 1/3$
- $p(x_3 = 1|x_4 = 0) = 1/6$ $p(x_3 = 1|x_4 = 1) = 5/6$

- $Likelihood(T) \sim = P(D|T) \sim = P(1011|T) P(1001|T)P(0100|T)$

- $P(1011|T) = p(x_4 = 1)p(x_1 = 1|x_4 = 1)p(x_2 = 0|x_4 = 1)p(x_3 = 1|x_4 = 1) = \frac{1}{2} \times \frac{1}{2} \times \frac{2}{3} \times \frac{5}{6} = \frac{10}{72}$

- $P(1001|T) = \frac{1}{2} \times \frac{1}{2} \times \frac{2}{3} \times \frac{5}{6} = \frac{10}{72}$

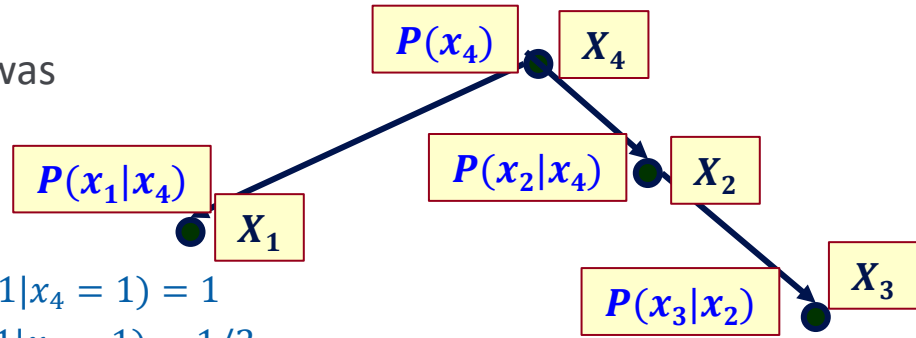
- $P(0100|T) = \frac{1}{2} \times \frac{1}{2} \times \frac{2}{3} \times \frac{5}{6} = \frac{10}{72}$

- $P(Data|Tree) = 125/4 * 3^6$

Example: Learning Distributions

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

- $P(x_4 = 1) = 2/3$
- $p(x_1 = 1|x_4 = 0) = 1/3$ $p(x_1 = 1|x_4 = 1) = 1$
- $p(x_2 = 1|x_4 = 0) = 1$ $p(x_2 = 1|x_4 = 1) = 1/2$
- $p(x_3 = 1|x_2 = 0) = 2/3$ $p(x_3 = 1|x_2 = 1) = 1/6$



- $Likelihood(T) \sim = P(D|T) \sim = P(1011|T)P(1001|T)P(0100|T)$

- $P(1011|T) = p(x_4 = 1)p(x_1 = 1|x_4 = 1)p(x_2 = 0|x_4 = 1)p(x_3 = 1|x_2 = 0) = \frac{2}{3} \times 1 \times \frac{1}{2} \times \frac{2}{3} = \frac{2}{9}$
- $P(1001|T) = \frac{1}{2} \times \frac{1}{2} \times \frac{2}{3} \times \frac{1}{6} = \frac{1}{36}$
- $P(0100|T) = \frac{1}{2} \times \frac{1}{2} \times \frac{1}{3} \times \frac{5}{6} = \frac{5}{72}$
- $P(Data|Tree) = 10/3^6 2^6$

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

Example: Summary

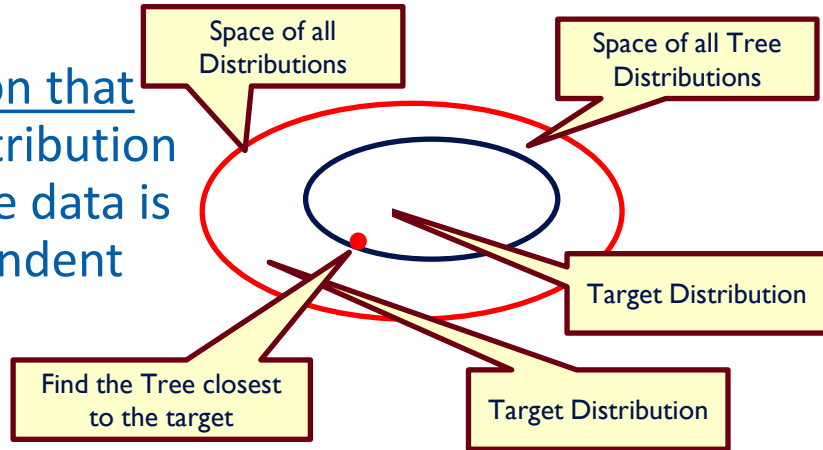
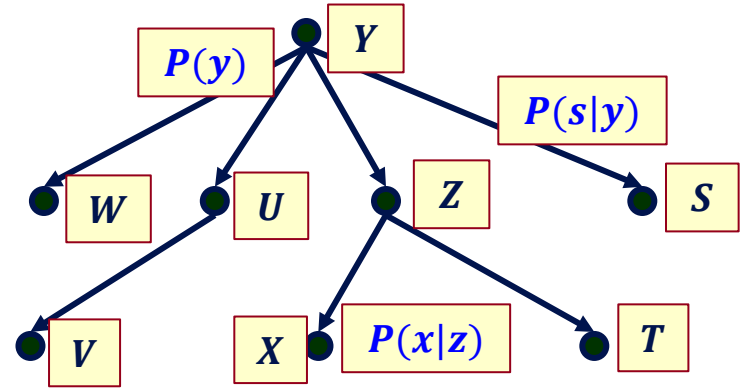
- We are now in the same situation we were when we decided which of two coins, fair (0.5,0.5) or biased (0.7,0.3) generated the data.
- **But, this isn't the most interesting case.**
- In general, we will not have a small number of possible distributions to choose from, but rather a parameterized family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

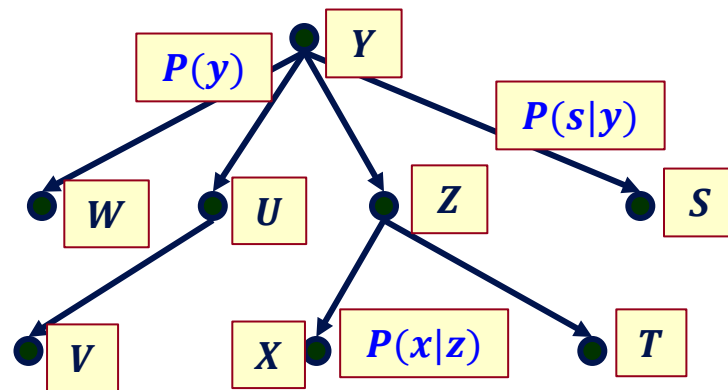
Learning Tree Dependent Distributions

- 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.)



Learning Tree Dependent Distributions

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

$$\begin{aligned} L(T) &= P(D|T) = \\ &= \operatorname{argmax}_T \prod_{\{x\}} P_T(x_1, x_2, \dots, x_n) = \\ &= \operatorname{argmax}_T \prod_{\{x\}} P_T(x_i | \text{Parents}(x_i)) \end{aligned}$$

(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):

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

- 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)

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

- 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.

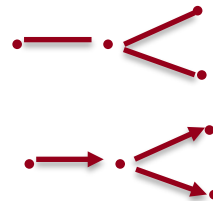
(2) • Let $I(x, y)$ be the weights of the edge (x, y)

- 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 .

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

- Recall:

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

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

$$\begin{aligned} D(P, T) &= \sum_x P(x) \log \frac{P(x)}{T(x)} = \\ &= \sum_x P(x) \log P(x) - \sum_x P(x) \log T(x) = \\ &= -H(x) - \sum_x P(x) \sum_{i=1}^n \log T(x_i | \pi(x_i)) = \end{aligned}$$

Slight abuse of notation at the root

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

Correctness (1)

$$\begin{aligned}
 D(P, T) &= \sum_x P(x) \log \frac{P(x)}{T(x)} = \sum_x P(x) \log P(x) - \sum_x P(x) \log T(x) = \\
 &= -H(x) - \sum_x P(x) \sum_{i=1}^n \log T(x_i | \pi(x_i)) = \\
 &= -H(x) - E_P \left[\sum_{i=1}^n \log T(x_i | \pi(x_i)) \right] = \\
 &= -H(x) - \sum_{i=1}^n E_P [\log T(x_i | \pi(x_i))] = \\
 &= -H(x) - \sum_{i=1}^n \sum_{(x_i, \pi(x_i))} P(x_i, \pi(x_i)) \log T(x_i | \pi(x_i)) = \\
 &= -H(x) - \sum_{i=1}^n \sum_{\pi(x_i)} P(\pi(x_i)) \sum_{x_i} P(x_i | \pi(x_i)) \log T(x_i | \pi(x_i))
 \end{aligned}$$

Definition of expectation:

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

To see this look at the difference:
 $\sum_i P(x_i | \pi(x_i)) \log P(x_i | \pi(x_i)) - \sum_i P(x_i | \pi(x_i)) \log T(x_i | \pi(x_i)) =$
 $= D(P(x_i | \pi(x_i)), T(x_i | \pi(x_i)))$
 This is non-negative, and equal to 0 only when $T=P$.

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

- However:

$$\log P(x_i | \pi(x_i)) = \log \frac{P(x_i, \pi(x_i))}{P(x_i)P(\pi(x_i))} + \log P(x_i)$$

$$P(x_i, \pi(x_i)) \log P(x_i | \pi(x_i)) = \underbrace{P(x_i, \pi(x_i)) \log \frac{P(x_i, \pi(x_i))}{P(x_i)P(\pi(x_i))}} + P(x_i, \pi(x_i)) \log P(x_i)$$

- This gives:

$$D(P, T) = -H(x) - \sum_1^n \underbrace{I(x_i, \pi(x_i))} - \sum_1^n \sum_x \underbrace{P(x_i)} \log P(x_i)$$

- 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.

$$D(P, T) = -H(x) - \sum_1^n I(x_i, \pi(x_i)) - \sum_1^n \sum_x P(x_i) \log P(x_i)$$

- 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 .

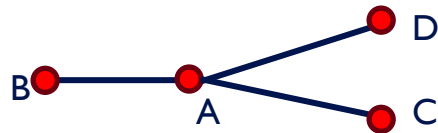
$$L(T) = P(D|T) = \prod_i P_T(x_i | \text{Parent}(x_i))$$

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

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



Learning tree Dependent Distributions

- Chow-Liu algorithm
- In particular, it learns D . (with some assumptions)
- Less is known about how fast it converges. (with some assumptions)
- Notice that we need to know the probabilities of some event in order to evaluate the likelihood.

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

CIS 419/519 Fall'19

54

- One may ask the question: why do we need this structure ? Why can't we answer the query directly from the data ?
- (Almost like making prediction directly from the data in the **badges problem**)

Administration (12/10/20)

Are we recording? YES!

Available on the web site

- Remember that all the lectures are available on the website **before the class**
 - **Go over it and be prepared**
 - **A new set** of written notes will accompany most lectures, with some more details, examples and, (when relevant) some code.
- **HW5** is out; due 12/10 (last day of the semester).
 - It is mostly a summary of the material we covered this semester (with a focus on the second half) and will help you prepare for the exam. No programming.
 - An **extension until Tuesday 12/15 – no additional slack day. This is a hard deadline.**
- The **Final** is on 12/18.
 - Similar style to the mid-term. Comprehensive, with emphasis on the material after the mid-term.
 - 90 minutes. You can start it any time between 9am and 7:30 pm ET. Your 90 minutes will be measure from the time you start.
 - Between 10:30-noon ET, most of the TAs will be on-board to respond to clarification questions. If possible, do it at that time.
 - Please open your video – if there are any technical issues, let me know.
 - There will be at least one TA available throughout the day to respond to clarification questions.
 - Communication will only be done via private Piazza posts.

▪ **Extra Time?**

Projects

- CIS 519 students need to do a team project: Read the [project descriptions](#) and follow the updates on the [Project webpage](#)
 - Teams will be of size 2-4
 - We will help grouping if needed
- There will be 3 options for projects.
 - Natural Language Processing (Text)
 - Computer Vision (Images)
 - Speech (Audio)
- In all cases, we will give you datasets and initial ideas
 - The problem will be multiclass classification problems
 - You will get annotated data only for some of the labels, but will also have to predict other labels
 - 0-zero shot learning; few-shot learning; transfer learning
- A detailed note will come out today.
- Timeline:
 - 11/11 Choose a project and team up
 - 11/23 Initial proposal describing what your team plans to do
 - 12/2 Progress report: 1 page. What you have done; plans; problems.
 - 12/21 Final paper + short video
- Try to make it interesting!

Review

- **Applied Machine Learning**
- **Applied:** mostly in HW
- **Machine learning:** mostly in class, quizzes, exams

Learning Theory

- PAC Learning
 - Occam’s Razor
 - Consistent Learners
 - Generalizing to “almost consistent”
- What does learnability depend on?
- From finite hypothesis space to infinite
 - VC Dimension

Agnostic Learning

- Assume we are trying to learn a concept f using hypotheses in H , but $f \notin H$
- In this case, our goal should be to find a hypothesis $h \in H$, with a small training error:

$$Error_{TR}(h) = \frac{1}{m} \sum_{\mathbf{x} \in \text{training examples}} |f(\mathbf{x}) - h(\mathbf{x})|$$

- We want a guarantee that a hypothesis with a small training error will have a good accuracy on unseen examples

$$Error_D(h) = \Pr_{\mathbf{x} \in D} [f(\mathbf{x}) \neq h(\mathbf{x})]$$

- We get a **generalization bound**—a bound on how much will the true error E_D deviate from the observed (training) error E_{TR} .
- For any distribution D generating training and test instances, with probability at least $1 - \delta$ over the choice of the training set of size m , (drawn IID), for all $h \in H$

$$Error_D < Error_{TR}(h) + \left[\frac{\log|H| + \log\left(\frac{1}{\delta}\right)}{2m} \right]^{\frac{1}{2}}$$

Generalization: a function of the Hypothesis class size

- See slide 102 in the On-line Lecture
- So, what should m be?

Agnostic Learning

- An agnostic learner
 - which makes no commitment to whether f is in H , and
- returns the hypothesis **with least** training error over at least the following number of examples m
- can guarantee with probability at least $(1 - \delta)$ that its **training error** is not off by more than ϵ from the **true error**.

$$m > \frac{1}{2\epsilon^2} \left\{ \ln(|H|) + \ln\left(\frac{1}{\delta}\right) \right\}$$

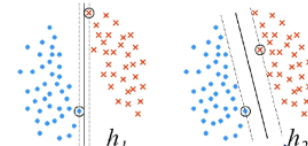
Learnability depends on the log of the size of the hypothesis space

SVMs

- Max Margin
 - Relations to VC dimension
 - Relations to $||W||$
- Support Vectors Machines
 - Hard SVM Optimization
 - Soft SVM Optimization
 - Regularization
 - Dual and Primal
 - Kernels
- Back to SGD
 - Relations to Perceptron
- Multiclass SVM
 - Notion of margin

Margin of a Separating Hyperplane

- A separating hyperplane: $\mathbf{w}^T \mathbf{x} + b = 0$



$$\begin{aligned} \mathbf{w}^T \mathbf{x}_i + b &\geq 1 && \text{if } y_i = 1 \\ \mathbf{w}^T \mathbf{x}_i + b &\leq -1 && \text{if } y_i = -1 \end{aligned}$$

$$\rightarrow y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

Distance between $\mathbf{w}^T \mathbf{x} + b = +1$ and -1 is $2/||\mathbf{w}||$

What we did:

1. Consider all possible \mathbf{w} with different angles
2. Scale \mathbf{w} such that the constraints are tight (closest points are on the ± 1 line)
3. Pick the one with largest margin/minimal size

Assumption: data is linearly separable
Let (x_0, y_0) be a point on $\mathbf{w}^T \mathbf{x} + b = 1$
Then its distance to the separating plane $\mathbf{w}^T \mathbf{x} + b = 0$ is: $|\mathbf{w}^T x_0 + b|/||\mathbf{w}|| = 1/||\mathbf{w}||$

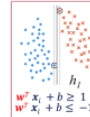
$$\begin{aligned} \mathbf{w}^T \mathbf{x} + b &= 0 \\ \mathbf{w}^T \mathbf{x} + b &= -1 \end{aligned}$$

CIS 419/519 Fall'20

17

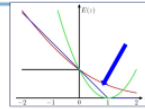
Soft SVM (2)

- Now, we want to solve:



$$\min_{\mathbf{w}, \xi} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_i \xi_i$$

$$\text{s.t. } \xi_i \geq 1 - y_i \mathbf{w}^T \mathbf{x}_i$$



$$\xi_i \geq 0 \quad \forall i$$

$$\text{In optimum, } \xi_i = \max(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i)$$

- Which can be written as:

$$\min_{\mathbf{w}} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_i \max(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i)$$

- What is the interpretation of this?

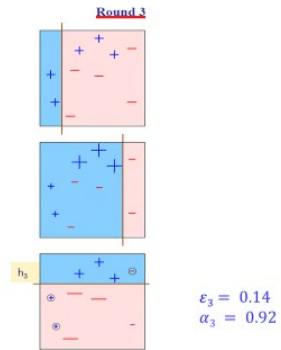
CIS 419/519 Fall'20

33

Boosting, Ensembles, Multiclass

- Weak vs. Strong Learnability
- Notion of Error
 - Distribution dependent
 - Changing the distribution of the data
 - Adaboost: proof and why does it work
- Boosting, Bagging, & Random Forests
- Multiclass
 - 1 vs. all
 - All vs. all
 - Global multiclass (SVM)

A Toy Example

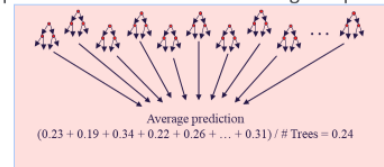


CIS 419/519 Fall'20

16

Random Forests (Bagged Trees++)

- Draw 1000 + bootstrap samples of data
- Draw sample of available attributes at each split
- Train trees on each sample/attribute set \rightarrow 1000 + trees
- Average prediction of trees on out-of-bag samples



CIS 419/519 Fall'20

32

NNs and Deep Learning

- Neural Networks
 - Expressivity
 - Non-linearity and differentiability
 - Backpropagation
 - Delta Rule
- Convolutional Networks
- Recurrent Neural Networks
 - Embeddings

The Backpropagation Algorithm

- Create a fully connected three layer network. Initialize weights.
- Until all examples produce the correct output within ϵ (or other criteria)

For each example in the training set do:

1. Compute the network output for this example
2. Compute the error between the output and target value
$$\delta_k = (t_k - o_k) o_k (1 - o_k)$$

1. For each output unit k , compute error term

$$\delta_j = o_j (1 - o_j) \cdot \sum_{k \in \text{downstream}(j)} -\delta_k w_{jk}$$

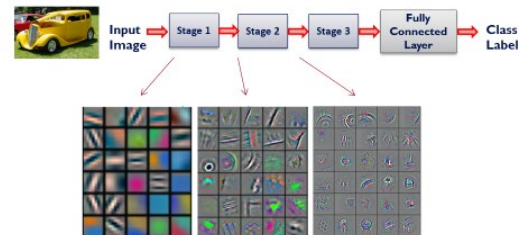
1. For each hidden unit, compute error term: $\Delta w_{ij} = R \delta_j x_i$
2. Update network weights with Δw_{ij}

End epoch

CIS 419/519 Fall'20

41

Convolutional Nets



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

CIS 419/519 Fall'20

72

Generative Models, Naïve Bayes

- Bayes Rule
- Generative Models
- Bayesian Learning Scenario
 - Examples
- Bayesian Classifiers
 - Naïve Bayes
 - Why does it work?
- Logistic Regression

Learning Scenario

$$P(h|D) = P(D|h)P(h)/P(D)$$

- The learner considers a set of candidate hypotheses H (models), and attempts to find the most probable one $h \in H$, given the observed data.
- Such maximally probable hypothesis is called maximum a posteriori hypothesis (MAP); Bayes theorem is used to compute it:

$$h_{MAP} = \operatorname{argmax}_{h \in H} P(h|D) = \operatorname{argmax}_{h \in H} P(D|h)P(h)/P(D) \\ = \operatorname{argmax}_{h \in H} P(D|h)P(h)$$

CIS 419/519 Fall'20

42

Recall: Naïve Bayes, Two Classes

- In the case of two classes we have:

$$\frac{P(v_j = 1|x)}{P(v_j = 0|x)} = \sum_i w_i x_i - b$$

- but since

$$P(v_j = 1|x) = 1 - P(v_j = 0|x)$$

- We showed that:

$$P(v_j = 1|x) = \frac{1}{1 + \exp(-\sum_i w_i x_i + b)}$$

- Which is simply the logistic (sigmoid) function used in the neural network representation.

CIS 419/519 Fall'20

94

Logistic Regression (3)

- Using the standard mapping to linear separators through the origin, we would like to minimize:
$$\min_{w,b} \sum_{i=1}^n \log P(y = +1 | x_i, w) = \min_{w,b} \sum_{i=1}^n \log [1 + \exp(-y_i(w^T x_i + b))]$$
- To get good generalization, it is common to add a regularization term, and the regularized logistics regression then becomes:

$$\min_w f(w) = \underbrace{\frac{1}{2} w^T w}_\text{Regularization term} + C \underbrace{\sum_{i=1}^m \log [1 + \exp(-y_i(w^T x_i))]}_\text{Empirical loss}$$

Where C is a user selected parameter that balances the two terms.

CIS 419/519 Fall'20

98

Un/Semi-Supervised Learning: EM and K-Means

- Semi-Supervised
 - Bootstrapping from a few labeled examples
 - Fractional Labels
- Expectation-Maximization
 - Coin problem
 - K-Means

Using Naïve Bayes

- For example, what can be done with the example (1000) ?
 - We have an estimate for its label...
 - But, can we use it to improve the classifier (that is, the estimation of the probabilities that we will use in the future)?
- Option 1: We can make predictions, and believe them
 - Or some of them (based on what?)
- Option 2: We can assume the example $x = (1000)$ is a
 - An n -labeled example with probability $\frac{P_n(x)}{P_n(x) + P_y(x)}$
 - A v -labeled example with probability $\frac{P_y(x)}{P_n(x) + P_y(x)}$
- Estimation of probabilities does not require working with integers!

What do we do once we have these labels?

We estimate the most likely parameters: $P(n), P(v); P(x|n), P(x|v)$
That is, we run NB again

CIS 419/519 Fall'20

10

Summary: EM Algorithm (Coins)

- We will assume (for a minute) that we know the parameters $\hat{p}, \hat{q}, \hat{\alpha}$ and use it to estimate which Coin is

$$P_1^i = P(\text{Coin1}|D^i) = \frac{P(D^i|\text{Coin1})P(\text{Coin1})}{P(D^i)} = \frac{\hat{\alpha} \hat{p}^h (1 - \hat{p})^{m-h}}{\hat{\alpha} \hat{p}^h (1 - \hat{p})^{m-h} + (1 - \hat{\alpha}) \hat{q}^h (1 - \hat{q})^{m-h}}$$

- STEP 2: Maximization Step

$$\frac{dE}{d\hat{\alpha}} = \sum_{i=1}^n \frac{P_1^i}{\hat{\alpha}} \frac{1 - P_1^i}{1 - \hat{\alpha}} = 0 \Rightarrow \hat{\alpha} = \frac{\sum P_1^i}{n}$$

$$\frac{dE}{d\hat{p}} = \sum_{i=1}^n P_1^i \left(\frac{h_i}{\hat{p}} - \frac{m - h_i}{1 - \hat{p}} \right) = 0 \Rightarrow \hat{p} = \frac{\sum P_1^i \frac{h_i}{m}}{\sum P_1^i}$$

$$\frac{dE}{d\hat{q}} = \sum_{i=1}^n (1 - P_1^i) \left(\frac{h_i}{\hat{q}} - \frac{m - h_i}{1 - \hat{q}} \right) = 0 \Rightarrow \hat{q} = \frac{\sum (1 - P_1^i) \frac{h_i}{m}}{\sum (1 - P_1^i)}$$

- Iterate

CIS 419/519 Fall'20

24

Summary: K-Means Algorithms

- Given a set $D = \{x_1, \dots, x_m\}$ of data points, **guess** initial parameters $\sigma, \mu_1, \mu_2, \dots, \mu_k$
- Compute (for all i, j)

$$p_{ij} = E[z_{ij}] = \frac{\exp[-\frac{1}{2\sigma^2}(x_i - \mu_j)^2]}{\sum_{k=1}^k \exp[-\frac{1}{2\sigma^2}(x_i - \mu_k)^2]}$$
- and a **new set of means**:

$$\mu_j = \frac{\sum_{i=1}^m E[z_{ij}]x_i}{\sum_{i=1}^m E[z_{ij}]}$$
- **repeat** to convergence

Difference now we place "fractional" points into clusters.
 p_{ij} is the fractional label

Recall: Standard K-Means clustering

- Guess k centers.
- Repeat:
 - Place each point in its center, based on distance.
 - Re-estimate centers for each cluster.
 - Re-place points

Notice that this algorithm will find the best k means in the sense of minimizing the sum of square distance.

The hard EM algorithm (threshold the distribution and keep the top option)

CIS 419/519 Fall'20

Bayesian Networks

- Representing Distributions
 - Independence assumptions
 - # of parameters
- Tree Dependent Distributions
 - Inference
 - Most likely distribution

Bayesian Network: Example

How many parameters do we have?
How many would we have if we had to store the entire joint on 6 variables, without any independence assumptions?

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

$$\begin{aligned}
 P(E, B, A, R, M, J) &= P(E) P(B, A, R, M, J | E) = \\
 &= P(E) P(B) P(A, R, M, J | E, B) = \\
 &= P(E) P(B) P(R | E, B) P(M, J, A | E, B) = \\
 &= P(E) P(B) P(R | E) P(M, J | A, E, B) P(A | E, B) = \\
 &= P(E) P(B) P(R | E) P(M | A) P(J | A) P(A | E, B)
 \end{aligned}$$

CIS 419/519 Fall'20 22

Example: Learning tree Dependent Distributions

- We have 3 data points that have been generated according to the target distribution: 1011; 1001; 0100
- We need to estimate some parameters:
- $P(A = 1) = \frac{2}{3}$, $P(B = 1) = \frac{1}{3}$, $P(C = 1) = \frac{1}{3}$, $P(D = 1) = \frac{2}{3}$
- For the values 00, 01, 10, 11 respectively, we have that:
 - $P(A, B) = 0; 1/3; 2/3; 0$ $P(A, B)/P(A)P(B) = 0; 3; 3/2; 0$ $I(A, B) \sim 9/2$
 - $P(A, C) = 1/3; 0; 1/3; 1/3$ $P(A, C)/P(A)P(C) = 3/2; 0; 3/4; 3/2$ $I(A, C) \sim 15/4$
 - $P(A, D) = 1/3; 0; 0; 2/3$ $P(A, D)/P(A)P(D) = 3; 0; 0; 3/2$ $I(A, D) \sim 9/2$
 - $P(B, C) = 1/3; 1/3; 1/3; 0$ $P(B, C)/P(B)P(C) = 3/4; 3/2; 3/2; 0$ $I(B, C) \sim 15/4$
 - $P(B, D) = 0; 2/3; 1/3; 0$ $P(B, D)/P(B)P(D) = 0; 3; 3/2; 0$ $I(B, D) \sim 9/2$
 - $P(C, D) = 1/3; 1/3; 0; 1/3$ $P(C, D)/P(C)P(D) = 3/2; 3/4; 0; 3/2$ $I(C, D) \sim 15/4$
- Generate the tree; place probabilities.

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

CIS 419/519 Fall'20 78

Example: Learning Distributions

- **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:**

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

CIS 419/519 Fall'20 56