



Lecture 32 of 42

Advanced Topics in Uncertain Reasoning Discussion: Relational Models, PS7

Wednesday, 08 November 2006

William H. Hsu
Department of Computing and Information Sciences, KSU

KSOL course page: <http://snipurl.com/v9v3>
Course web site: <http://www.kddresearch.org/Courses/Fall-2006/CIS730>
Instructor home page: <http://www.cis.ksu.edu/~bhsu>

Reading for Next Class:
Chapter 15, Russell & Norvig 2nd edition



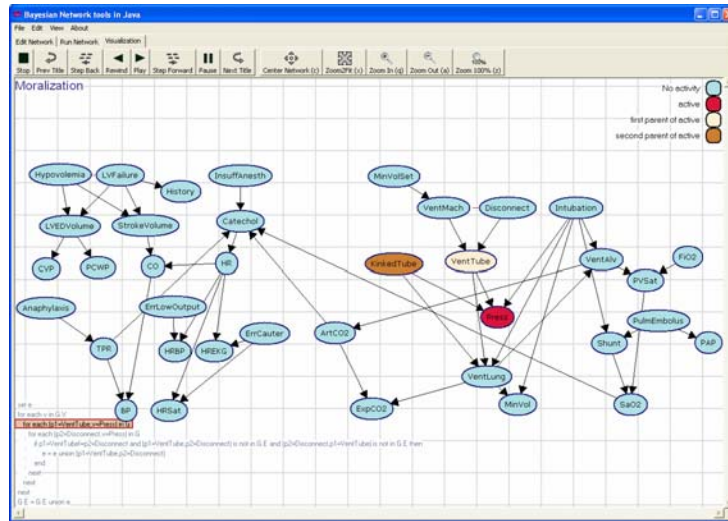
Lecture Outline

- Wednesday's Reading: Sections 14.6 – 14.8, R&N 2e, Chapter 15
- Friday: Sections 18.1 – 18.2, R&N 2e
- Today: Inference in Graphical Models
 - * Bayesian networks and causality
 - * Inference and learning
 - * BNJ interface (<http://bnj.sourceforge.net>)
 - * Causality





BNJ Visualization: Pseudo-Code Annotation (Code Page)



ALARM
Network

© 2004 KSU BNJ Development Team

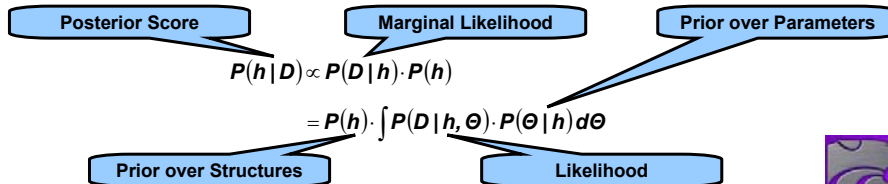


Other Topics in Graphical Models [2]: Learning Structure from Data

- General-Case BBN Structure Learning: *Use Inference to Compute Scores*
- Optimal Strategy: Bayesian Model Averaging
 - * Assumption: models $h \in H$ are *mutually exclusive and exhaustive*
 - * *Combine predictions of models in proportion to marginal likelihood*
 - Compute conditional probability of hypothesis h given observed data D
 - *i.e.*, compute expectation over unknown h for unseen cases
 - Let $h =$ structure, parameters $\Theta \equiv$ CPTs

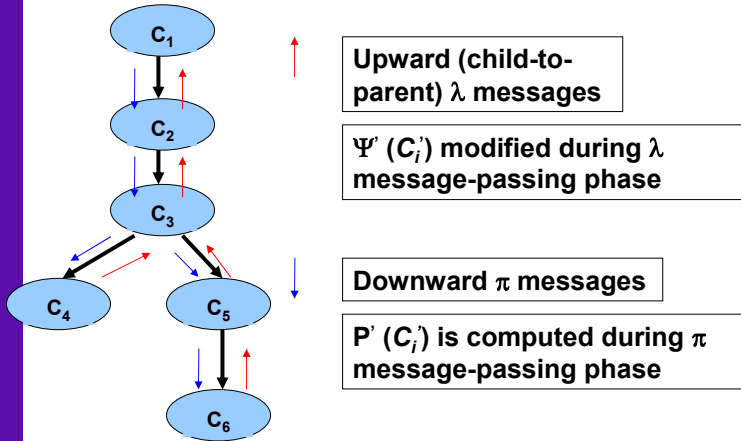
$$P(\bar{x}^{(m+1)} | D) = P(x_1, x_2, \dots, x_n | \bar{x}^{(1)}, \bar{x}^{(2)}, \dots, \bar{x}^{(m)})$$

$$= \sum_{h \in H} P(\bar{x}^{(m+1)} | D, h) \cdot P(h | D)$$





Propagation Algorithm in Singly-Connected Bayesian Networks – Pearl (1983)

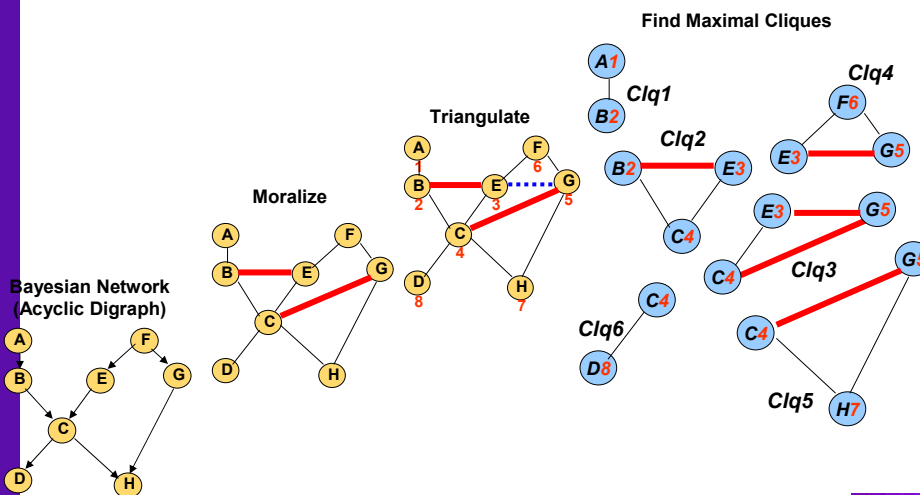


Multiply-connected case: exact, approximate inference are $\#P$ -complete (counting problem is $\#P$ -complete iff decision problem is NP -complete)

Adapted from Neapolitan (1990), Guo (2000)



Inference by Clustering [1]: Graph Operations (Moralization, Triangulation, Maximal Cliques)



Adapted from Neapolitan (1990), Guo (2000)



Inference by Clustering [2]: Function Tree – Lauritzen & Spiegelhalter (1988)

Input: list of cliques of triangulated, moralized graph G_u

Output:

Tree of cliques

Separator nodes S_i ,

Residual nodes R_i and potential probability $\Psi(\text{Clq}_i)$ for all cliques

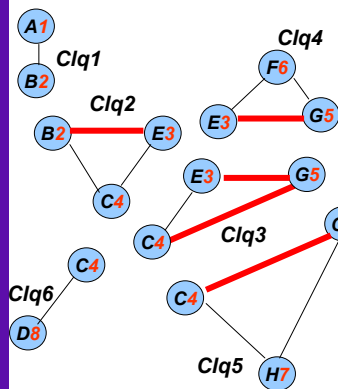
Algorithm:

1. $S_i = \text{Clq}_i \cap (\text{Clq}_1 \cup \text{Clq}_2 \cup \dots \cup \text{Clq}_{i-1})$
2. $R_i = \text{Clq}_i - S_i$
3. If $i > 1$ then identify a $j < i$ such that Clq_j is a parent of Clq_i
4. Assign each node v to a unique clique Clq_i that $v \cup c(v) \subseteq \text{Clq}_i$
5. Compute $\Psi(\text{Clq}_i) = \prod_{v \in \text{Clq}_i} P(v | c(v))$ {1 if no v is assigned to Clq_i }
6. Store Clq_i , R_i , S_i , and $\Psi(\text{Clq}_i)$ at each vertex in the tree of cliques

Adapted from Neapolitan (1990), Guo (2000)



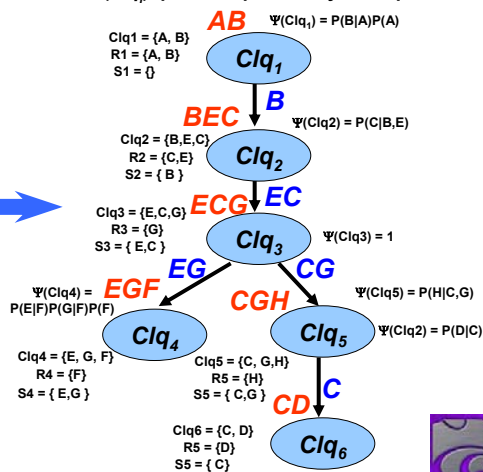
Inference by Clustering [3]: Clique-Tree Operations



R_i : residual nodes

S_i : separator nodes

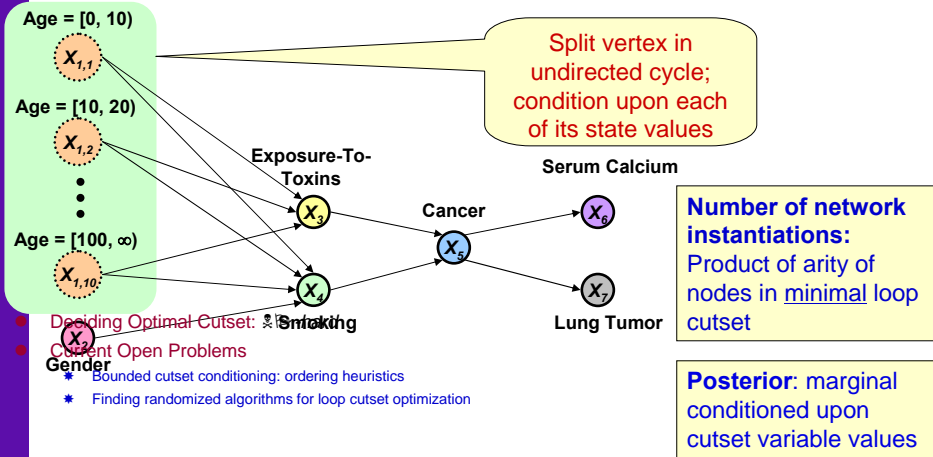
$\Psi(\text{Clq}_i)$: potential probability of Clique i



Adapted from Neapolitan (1990), Guo (2000)



Inference by Loop Cutset Conditioning



Inference by Variable Elimination [1]: Intuition

Enumeration is inefficient: repeated computation

e.g., computes $P(J = true|a)P(M = true|a)$ for each value of e

Variable elimination: carry out summations right-to-left, storing intermediate results (factors) to avoid recomputation

$$\begin{aligned}
 P(B|J = true, M = true) &= \alpha \underbrace{P(B)}_B \sum_e \underbrace{P(e)}_E \sum_a \underbrace{P(a|B, e)}_A \underbrace{P(J = true|a)}_J \underbrace{P(M = true|a)}_M \\
 &= \alpha P(B) \sum_e P(e) \sum_a P(a|B, e) P(J = true|a) f_M(a) \\
 &= \alpha P(B) \sum_e P(e) \sum_a P(a|B, e) f_J(a) f_M(a) \\
 &= \alpha P(B) \sum_e P(e) \sum_a f_A(a, b, e) f_J(a) f_M(a) \\
 &= \alpha P(B) \sum_e P(e) f_{\bar{A}JM}(b, e) \text{ (sum out } A) \\
 &= \alpha P(B) f_{\bar{E}\bar{A}JM}(b) \text{ (sum out } E) \\
 &= \alpha f_B(b) \times f_{\bar{E}\bar{A}JM}(b)
 \end{aligned}$$



Inference by Variable Elimination [2]: Factoring Operations

Pointwise product of factors f_1 and f_2 :

$$f_1(x_1, \dots, x_j, y_1, \dots, y_k) \times f_2(y_1, \dots, y_k, z_1, \dots, z_l) \\ = f(x_1, \dots, x_j, y_1, \dots, y_k, z_1, \dots, z_l)$$

E.g., $f_1(a, b) \times f_2(b, c) = f(a, b, c)$

Summing out a variable from a product of factors: move any constant factors outside the summation:

$$\sum_x f_1 \times \dots \times f_k = f_1 \times \dots \times f_i \sum_x f_{i+1} \times \dots \times f_k = f_1 \times \dots \times f_i \times f_{\bar{x}}$$

assuming f_1, \dots, f_i do not depend on X

Adapted from slides by S. Russell, UC Berkeley

<http://aima.cs.berkeley.edu/>

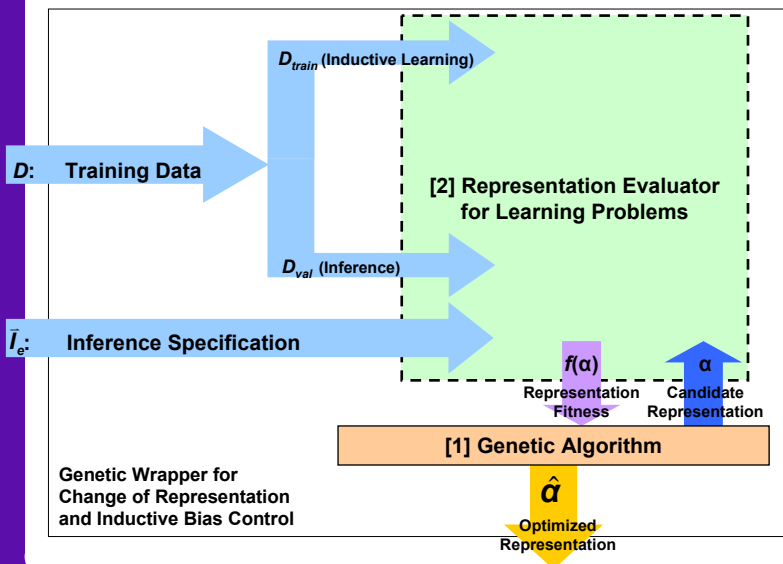
CIS 490 / 730: Artificial Intelligence

Wednesday, 08 Nov 2006

Computing & Information Sciences
Kansas State University



Genetic Algorithms for Parameter Tuning in Bayesian Network Structure Learning



CIS 490 / 730: Artificial Intelligence

Wednesday, 08 Nov 2006

Computing & Information Sciences
Kansas State University



Using Graphical Models

- **A Graphical View of Simple (Naïve) Bayes**

- * $x_i \in \{0, 1\}$ for each $i \in \{1, 2, \dots, n\}$; $y \in \{0, 1\}$

- * Given: $P(x_i | y)$ for each $i \in \{1, 2, \dots, n\}$; $P(y)$

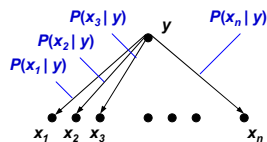
- * Assume **conditional independence**

- $\forall i \in \{1, 2, \dots, n\} \Rightarrow P(x_i | x_{\setminus i}, y) \equiv P(x_i | x_1, x_2, \dots, x_{i-1}, x_{i+1}, x_{i+2}, \dots, x_n, y) = P(x_i | y)$

- NB: this assumption entails the Naïve Bayes assumption

- Why? $P(x_1, x_2, \dots, x_n | y) = \prod_i P(x_i | x_{\setminus i}, y) = \prod_i P(x_i | y)$

- * Can compute $P(y | x)$ given this info



- * Can also compute the joint pdf over all $n + 1$ variables

$$P(\vec{x}, y) = P(y)P(\vec{x} | y) = P(y) \prod_{i=1}^n P(x_i | x_{\setminus i}, y) = P(y) \prod_{i=1}^n P(x_i | y)$$

- **Inference Problem for a (Simple) Bayesian Network**

- * Use the above model to compute the probability of any *conditional event*

- * Exercise: $P(x_1, x_2, y | x_3, x_4)$



In-Class Exercise: Probabilistic Inference

- **Inference Problem for a (Simple) Bayesian Network**

- * Model: Naïve Bayes

- * Objective: compute the probability of any *conditional event*

- **Exercise**

- * Given

- $P(x_i | y)$, $i \in \{1, 2, 3, 4\}$

- $P(y)$

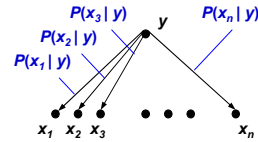
- * Want: $P(x_1, x_2, y | x_3, x_4)$

$$\begin{aligned} P(x_1, x_2, y | x_3, x_4) &= \frac{P(x_3, x_4 | x_1, x_2, y)P(x_1, x_2, y)}{P(x_3, x_4)} \\ &= \frac{P(x_1, x_2, x_3, x_4, y)}{P(x_3, x_4)} \\ &= \frac{P(y) \prod_{i=1}^4 P(x_i | y)}{\sum_y P(x_3 | y)P(x_4 | y)} \end{aligned}$$



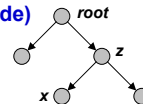
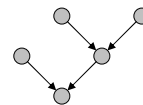
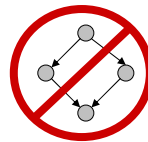
Unsupervised Learning and Conditional Independence

- **Given: $(n + 1)$ -Tuples** $(x_1, x_2, \dots, x_n, x_{n+1})$
 - * No notion of instance variable or label
 - * After seeing some examples, want to know something about the domain
 - Correlations among variables
 - Probability of certain events
 - Other properties
- **Want to Learn: Most Likely Model that Generates Observed Data**
 - * In general, a very hard problem
 - * *Under certain assumptions*, have shown that we can do it
- **Assumption: Causal Markovity**
 - * Conditional independence among “effects”, given “cause”
 - * *When is the assumption appropriate?*
 - * *Can it be relaxed?*
- **Structure Learning**
 - * Can we learn more general probability distributions?
 - * Examples: automatic speech recognition (ASR), natural language, etc.



Tree Dependent Distributions

- **Polytrees**
 - * *aka singly-connected Bayesian networks*
 - * **Definition:** a Bayesian network with no undirected loops
 - * **Idea:** restrict distributions (CPTs) to single nodes
 - * **Theorem:** inference in singly-connected BBN requires linear time
 - Linear in network size, including CPT sizes
 - Much better than for unrestricted (multiply-connected) BBNs
- **Tree Dependent Distributions**
 - * Further restriction of polytrees: every node has at one parent
 - * Now only need to keep 1 prior, $P(\text{root})$, and $n - 1$ CPTs (1 per node)
 - * All CPTs are 2-dimensional: $P(\text{child} | \text{parent})$
- **Independence Assumptions**
 - * As for general BBN: x is independent of non-descendants given (single) parent z
 - * *Very strong assumption* (applies in some domains but not most)

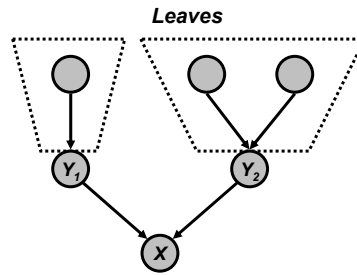




Inference in Trees

- **Inference in Tree-Structured BBNs ("Trees")**
 - * Generalization of Naïve Bayes to model of *tree dependent distribution*
 - * Given: tree T with all associated probabilities (CPTs)
 - * Evaluate: probability of a specified event, $P(x)$
- **Inference Procedure for Polytrees**
 - * Recursively traverse tree
 - Breadth-first, **source(s) to sink(s)**
 - Stop when query value $P(x)$ is known
 - * Perform inference at each node

$$\begin{aligned}
 P(x) &= P(X=x) \\
 &= \sum_{y_1, y_2} P(x | y_1, y_2) \cdot P(y_1, y_2) \\
 &= \sum_{y_1, y_2} P(x | y_1, y_2) \cdot P(y_1) \cdot P(y_2)
 \end{aligned}$$



parents(Y_1) = parents(Y_2) = X

- * **NB:** for *trees*, proceed **root to leaves** (e.g., breadth-first or depth-first)
- * Simple application of Bayes's rule (more efficient algorithms exist)



Learning Tree Distributions: Example

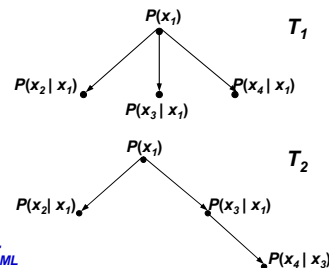
- **Candidate Models: Tree-Structured BBNs**
- **Learning Problem**
 - * Given: sample $D \sim$ distribution D
 - * Return: *most likely tree* T that generated D
 - * i.e., search for MAP hypothesis
- **MAP Estimation over BBNs**
 - * Assuming uniform priors on trees, $h_{MAP} = h_{ML} \equiv T_{ML}$

$$T_{ML} = \underset{T \in \mathcal{H}}{\operatorname{argmax}} P(D|T)$$

- * Maximization program

$$\begin{aligned}
 T_{ML} &= \underset{T \in \mathcal{H}}{\operatorname{argmax}} \prod_{(x_1, x_2, \dots, x_n) \in D} P_T(x_1, x_2, \dots, x_n) \\
 &= \underset{T \in \mathcal{H}}{\operatorname{argmax}} \prod_{(y_1, \dots, y_n) \in D} \prod_i P_T(x_i | \text{parents}(x_i))
 \end{aligned}$$

- * Try this for Naïve Bayes...

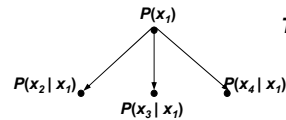




In-Class Exercise: Learning Distributions [1]

- Input: $D \equiv \{1011, 1001, 0100\} \sim D$
- CPT for P_1

x	$P(x)$	x	$P(x)$	x	$P(x)$	x	$P(x)$
0000	0.1	0100	0.1	1000	0.0	1100	0.05
0001	0.1	0101	0.1	1001	0.0	1101	0.05
0010	0.1	0110	0.1	1010	0.0	1110	0.05
0011	0.1	0111	0.1	1011	0.0	1111	0.05



- CPT for P_2

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

- CPT for P_3

$P(x_1 = 1) = 2/3$	
$P(x_2 = 1 x_1 = 0) = 1$	$P(x_2 = 1 x_1 = 1) = 0$
$P(x_3 = 1 x_1 = 0) = 0$	$P(x_3 = 1 x_1 = 1) = 1/2$
$P(x_4 = 1 x_1 = 0) = 0$	$P(x_4 = 1 x_1 = 1) = 1$

- Candidate Models (Trees): $h_1 \equiv P_1$, $h_2 \equiv T(P_2)$, $h_3 \equiv T(P_3)$
- Tree-Structured BBN Learning



In-Class Exercise: Learning Tree Distributions [2]

- Input Data: $D \equiv \{1011, 1001, 0100\}$
- Candidate Models: CPTs for Table (T_1), T_2 , T_3
- Results: Likelihood Estimation

- * $P(D | T_1) = P(1011 | T_1) \cdot P(1001 | T_1) \cdot P(0100 | T_1) = 0.0 \cdot 0.0 \cdot 0.1 = 0.0$
- * $P(D | T_2) = P(1011 | T_2) \cdot P(1001 | T_2) \cdot P(0100 | T_2)$
 - $P(1011 | T_2) = P(x_4 = 1) \cdot P(x_3 = 1 | x_4 = 1) \cdot P(x_2 = 0 | x_4 = 1) \cdot P(x_1 = 1 | x_4 = 1) = 1/2 \cdot 1/2 \cdot 1/3 \cdot 5/6 = 5/72$
 - $P(1001 | T_2) = 1/2 \cdot 1/2 \cdot 2/3 \cdot 5/6 = 10/72$
 - $P(0100 | T_2) = 1/2 \cdot 1/2 \cdot 2/3 \cdot 5/6 = 10/72$
 - $P(D | T_2) = 500 / 373248 \approx 0.0013$
- * $P(D | T_3) = 1/27$
- * Likelihood (T_1) < Likelihood (T_2) < Likelihood (T_3)

- Notes

- * Conclusion: of candidate models, T_3 is most likely to have produced D
- * Looked at 3 fixed distributions (NB and T_1 , a tree with fixed structure)





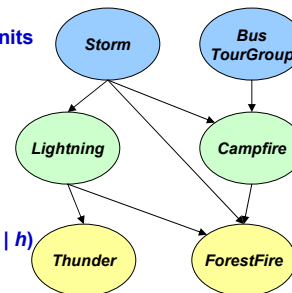
Learning Distributions: Objectives

- **Learning The Target Distribution**
 - * What is the target distribution?
 - * Can't use "the" target distribution
 - Case in point: suppose target distribution was P_t (collected over 20 examples)
 - Using Naïve Bayes would not produce an h close to the MAP/ML estimate
 - * Relaxing CI assumptions: expensive
 - MLE becomes intractable; BOC approximation, *highly* intractable
 - Instead, *should make judicious CI assumptions*
 - * As before, goal is *generalization*
 - Given D (e.g., {1011, 1001, 0100})
 - Would like to know $P(1111)$ or $P(11^{**}) \equiv P(x_1 = 1, x_2 = 1)$
- **Several Variants**
 - * Known or unknown structure
 - * Training examples may have missing values
 - * *Known structure and no missing values*: as easy as training Naïve Bayes



Learning Bayesian Networks: Partial Observability

- **Suppose Structure Known, Variables Partially Observable**
 - * **Example**
 - Can observe *ForestFire, Storm, BusTourGroup, Thunder*
 - Can't observe *Lightning, Campfire*
 - * Similar to training artificial neural net with hidden units
 - **Causes:** *Storm, BusTourGroup*
 - **Observable effects:** *ForestFire, Thunder*
 - **Intermediate variables:** *Lightning, Campfire*
- **Learning Algorithm**
 - * Can use gradient learning (as for ANNs)
 - * Converge to network h that (locally) maximizes $P(D | h)$
- **Analogy: Medical Diagnosis**
 - * **Causes:** diseases or diagnostic findings
 - * **Intermediates:** hidden causes or hypothetical inferences (e.g., heart rate)
 - * **Observables:** measurements (e.g., from medical instrumentation)





Learning Bayesian Networks: Gradient Ascent

- **Algorithm Train-BN (D)**

- * Let w_{ijk} denote one entry in the CPT for variable Y_i in the network
 - $w_{ijk} = P(Y_i = y_{ij} \mid \text{parents}(Y_i) = \langle \text{the list } u_{ik} \text{ of values} \rangle)$
 - e.g., if $Y_i = \text{Campfire}$, then (for example) $u_{ik} = \langle \text{Storm} = T, \text{BusTourGroup} = F \rangle$
- * WHILE termination condition not met DO // perform gradient ascent
 - Update all CPT entries w_{ijk} using training data D

$$w_{ijk} \leftarrow w_{ijk} + r \sum_{\mathbf{x}} \frac{\partial \log P(\mathbf{y}, \mathbf{u}_{ik} \mid \mathbf{x})}{\partial w_{ijk}}$$

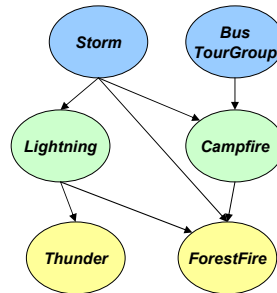
- **Renormalize w_{ijk} to assure invariants:**

$$\sum_j w_{ijk} = 1$$

$$\forall j. 0 \leq w_{ijk} \leq 1$$

- **Applying Train-BN**

- * Learns CPT values
- * Useful in case of *known structure*
- * Next: learning structure from data



Tree Dependent Distributions: Learning The Structure

- **Problem Definition: Find Most Likely T Given D**
- **Brute Force Algorithm**

- * FOR each tree T DO
 Compute the likelihood of T :

$$P(T \mid D) \propto P(D \mid T) = \underset{T \in \mathcal{H}}{\text{argmax}} \prod_{(x_1, x_2, \dots, x_n) \in D} \prod_i P_T(x_i \mid \text{parents}(x_i))$$

- * RETURN the maximal T

- **Is This Practical?**

- * Typically not... ($|\mathcal{H}|$ analogous to that of ANN weight space)
- * What can we do about it?

- **Solution Approaches**

- * Use criterion (scoring function): Kullback-Leibler (K-L) distance

$$D(P \parallel P') = \sum_x P(x) \log \frac{P(x)}{P'(x)}$$

- * Measures how well a distribution P approximates a distribution P'
- * aka K-L divergence, aka cross-entropy, aka relative entropy



Tree Dependent Distributions: Maximum Weight Spanning Tree (MWST)

- Input: m Measurements (n -Tuples), i.i.d. $\sim P$
- Algorithm *Learn-Tree-Structure* (D)
 - * FOR each variable X DO estimate $P(x)$ // binary variables: n numbers
 - * FOR each pair (X, Y) DO estimate $P(x, y)$ // binary variables: n^2 numbers
 - * FOR each pair DO compute the mutual information (measuring the information X gives about Y) with respect to this empirical distribution

$$I(X; Y) \equiv \sum_{x,y} P(x,y) \lg \frac{P(x,y)}{P(x) \cdot P(y)} = D(P(X,Y) || P(X) \cdot P(Y))$$

- * Build a complete undirected graph with all the variables as vertices
- * Let $I(X; Y)$ be the weight of edge (X, Y)
- * Build a Maximum Weight Spanning Tree (MWST)
- * Transform the resulting undirected tree into a directed tree (choose a root, and set the direction of all edges away from it)
- * Place the corresponding CPTs on the edges (gradient learning)
- * RETURN: a tree-structured BBN with CPT values



Tree Dependent Distributions: Example

- Input: $D \equiv \{1011, 1001, 0100\}$
- Estimation of Model Parameters
 - * $P(x_1 = 1) = 2/3, P(x_2 = 1) = 1/3, P(x_3 = 1) = 1/3, P(x_4 = 1) = 2/3$
 - * Pairs: 00, 01, 10, 11

• $P(x_1, x_2) = 0; 1/3; 2/3; 0$	$P(x_1, x_2) / P(x_1) \cdot P(x_2) = 0; 3; 3/2; 0$
• $P(x_1, x_3) = 1/3; 0; 1/3; 1/3$	$P(x_1, x_3) / P(x_1) \cdot P(x_3) = 3/2; 0; 3/4; 3/2$
• $P(x_1, x_4) = 1/3; 0; 0; 2/3$	$P(x_1, x_4) / P(x_1) \cdot P(x_4) = 3; 0; 0; 3/2$
• $P(x_2, x_3) = 1/3; 1/3; 1/3; 0$	$P(x_2, x_3) / P(x_2) \cdot P(x_3) = 3/4; 3/2; 3/2; 0$
• $P(x_2, x_4) = 0; 2/3; 1/3; 0$	$P(x_2, x_4) / P(x_2) \cdot P(x_4) = 0; 3; 3/2; 0$
• $P(x_3, x_4) = 1/3; 1/3; 0; 1/3$	$P(x_3, x_4) / P(x_3) \cdot P(x_4) = 3/2; 3/4; 0; 3/2$
 - * Use these CPTs as input to MWST and gradient learning
- MWST Algorithms
 - * MWST algorithms: Kruskal's algorithm, Prim's algorithm (quick sketch next time)
 - * Complexity: $O(n^2 \lg n)$
 - * See [Cormen, Leiserson, and Rivest, 1990]





Applications of Bayesian Networks

- **Inference: Decision Support Problems**

- * **Diagnosis**

- Medical [Heckerman, 1991]
 - Equipment failure

- * **Pattern recognition**

- Image identification: faces, gestures
 - Automatic speech recognition
 - Multimodal: [speechreading](#), emotions

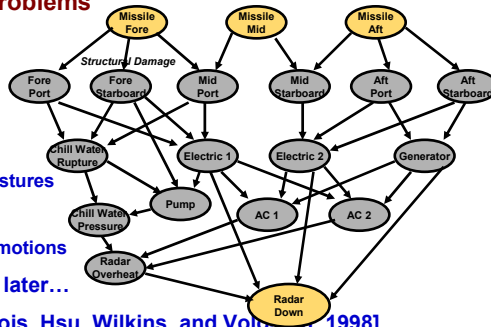
- * **Prediction: more applications later...**

- * **Simulation-based training [Grois, Hsu, Wilkins, and Volosin, 1998]**

- * **Control automation**

- Navigation with a mobile robot
 - Battlefield reasoning [Mengshoel, Goldberg, and Wilkins, 1998]

- **Learning: Acquiring Models for Inferential Applications**



Related Work in Bayesian Networks

- **BBN Variants, Issues Not Covered Yet**

- * **Temporal models**

- [Markov Decision Processes \(MDPs\)](#)
 - [Partially Observable Markov Decision Processes \(POMDPs\)](#)
 - Useful in reinforcement learning

- * **Influence diagrams**

- Decision theoretic model
 - Augments BBN with utility values and decision nodes

- * **Unsupervised learning (EM, *AutoClass*)**

- * **[Feature \(subset\) selection](#): finding relevant attributes**

- **Current Research Topics Not Addressed in This Course**

- * **[Hidden variables](#)** (introduction of new variables not observed in data)
 - * **[Incremental BBN learning](#)**: modifying network structure online ("on the fly")
 - * **Structure learning for stochastic processes**
 - * **[Noisy-OR](#) Bayesian networks**: another simplifying restriction



Tools for Building Graphical Models

- Commercial Tools: *Ergo*, *Netica*, *TETRAD*, *Hugin*
- *Bayes Net Toolbox (BNT)* – Murphy (1997-present)
 - * Distribution page <http://http.cs.berkeley.edu/~murphyk/Bayes/bnt.html>
 - * Development group <http://groups.yahoo.com/group/BayesNetToolbox>
- *Bayesian Network tools in Java (BNJ)* – Hsu *et al.* (1999-present)
 - * Distribution page <http://bnj.sourceforge.net>
 - * Development group <http://groups.yahoo.com/group/bndev>
 - * Current (re)implementation projects for KSU KDD Lab
 - *Continuous state*: Minka (2002) – Hsu, Guo, Li
 - Formats: XML BNIF (MSBN), Netica – Barber, Guo
 - Space-efficient DBN inference – Meyer
 - Bounded cutset conditioning – Chandak



References: Graphical Models and Inference Algorithms

- **Graphical Models**
 - * **Bayesian (Belief) Networks tutorial** – Murphy (2001)
<http://www.cs.berkeley.edu/~murphyk/Bayes/bayes.html>
 - * **Learning Bayesian Networks** – Heckerman (1996, 1999)
<http://research.microsoft.com/~heckerman>
- **Inference Algorithms**
 - * **Junction Tree (Join Tree, L-S, *Hugin*)**: Lauritzen & Spiegelhalter (1988)
<http://citeseer.nj.nec.com/huang94inference.html>
 - * **(Bounded) Loop Cutset Conditioning**: Horvitz & Cooper (1989)
<http://citeseer.nj.nec.com/shachter94global.html>
 - * **Variable Elimination (Bucket Elimination, *ElimBel*)**: Dechter (1986)
<http://citeseer.nj.nec.com/dechter96bucket.html>
 - * **Recommended Books**
 - Neapolitan (1990) – *out of print*; see [Pearl \(1988\)](#), Jensen (2001)
 - Castillo, Gutierrez, Hadi (1997)
 - Cowell, Dawid, Lauritzen, Spiegelhalter (1999)
 - * **Stochastic Approximation**
<http://citeseer.nj.nec.com/cheng00aisbn.html>



Terminology

- Introduction to Reasoning under Uncertainty
 - * Probability foundations
 - * Definitions: subjectivist, frequentist, logistic
 - * (3) Kolmogorov axioms
- Bayes's Theorem
 - * Prior probability of an event
 - * Joint probability of an event
 - * Conditional (posterior) probability of an event
- Maximum *A Posteriori* (MAP) and Maximum Likelihood (ML) Hypotheses
 - * MAP hypothesis: highest conditional probability given observations (data)
 - * ML: highest likelihood of generating the observed data
 - * ML estimation (MLE): estimating parameters to find ML hypothesis
- Bayesian Inference: Computing Conditional Probabilities (CPs) in A Model
- Bayesian Learning: Searching Model (Hypothesis) Space using CPs



Summary Points

- Introduction to Probabilistic Reasoning
 - * Framework: using probabilistic criteria to search H
 - * Probability foundations
 - ⇒ Definitions: subjectivist, objectivist; Bayesian, frequentist, logistic
 - ⇒ Kolmogorov axioms
- Bayes's Theorem
 - * Definition of conditional (posterior) probability
 - * Product rule
- Maximum *A Posteriori* (MAP) and Maximum Likelihood (ML) Hypotheses
 - * Bayes's Rule and MAP
 - * Uniform priors: allow use of MLE to generate MAP hypotheses
 - * Relation to version spaces, candidate elimination
- Next Week: Chapter 14, Russell and Norvig
 - * Later: Bayesian learning: MDL, BOC, Gibbs, Simple (Naïve) Bayes
 - * Categorizing text and documents, other applications