Lecture 12

Introduction to Bayesian Networks

Tuesday, October 5, 1999

William H. Hsu
Department of Computing and Information Sciences, KSU
http://www.cis.ksu.edu/~bhsu

Readings:
Section 6.11, Mitchell
Chapter 15, Russell and Norvig
"Bayesian Networks Without Tears", Charniak

Lecture Outline

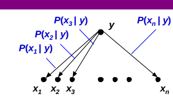
- Readings: 6.11, Mitchell; Chapter 15, Russell and Norvig; Charniak Tutorial
- Suggested Exercises: 6.6, Mitchell; 15.2 Russell and Norvig
- This Week's Review: "A Theory of Inferred Causation", Pearl and Verma
- Graphical Models of Probability
  - Bayesian networks: introduction
    - Definition and basic principles
    - Conditional independence and causal Markovity
  - Inference and learning using Bayesian networks
    - Acquiring and applying distributions (conditional probability tables)
    - Learning tree dependent distributions and polytrees
- Learning Distributions for Networks with Specified Structure
  - Gradient learning
  - Maximum weight spanning tree (MWST) algorithm for tree-structured networks
- Reasoning under Uncertainty: Applications and Augmented Models
- Next Lecture: (More on) Learning Bayesian Network Structure

Graphical Models of Probability Distributions

- Idea
  - Want: model that can be used to perform inference
  - Desired properties
    - Ability to represent functional, logical, stochastic relationships
    - Express uncertainty
    - Observe the laws of probability
    - Tractable inference when possible
    - *Can be learned from data*
- Additional Desiderata
  - Ability to incorporate knowledge
    - Knowledge acquisition and elicitation: in format *familiar to domain experts*
    - Language of subjective probabilities and relative probabilities
  - Support decision making
    - Represent utilities (cost or value of information, state)
    - Probability theory + utility theory = decision theory
  - Ability to reason over time (temporal models)

Using Graphical Models

- A Graphical View of Simple (Naïve) Bayes



  - $x_i \in \{0, 1\}$ for each $i \in \{1, 2, \ldots, n\}$; $y \in \{0, 1\}$
  - Given: $P(x_i \mid y)$ for each $i \in \{1, 2, \ldots, n\}$; $P(y)$
  - Assume conditional independence
    - $\forall\, i \in \{1, 2, \ldots, n\} \Rightarrow P(x_i \mid x_{\neq i}, y) \equiv P(x_i \mid x_1, x_2, \ldots, x_{i-1}, x_{i+1}, x_{i+2}, \ldots, x_n, y) = P(x_i \mid y)$
    - *NB: this assumption entails the Naïve Bayes assumption*
    - Why? $P(x_1, x_2, \ldots, x_n \mid y) = \prod_i P(x_i \mid x_{\neq i}, y) = \prod_i P(x_i \mid y)$
  - Can compute $P(y \mid x)$ given this info

$$P(y \mid \vec{x}) = \frac{P(\vec{x} \mid y)P(y)}{P(\vec{x})} = \frac{P(y)}{P(\vec{x})}\prod_{i=1}^{n} P(x_i \mid x_{\neq i}, y) = \frac{P(y)}{P(\vec{x})}\prod_{i=1}^{n} P(x_i \mid y)$$

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

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

- Inference Problem for a (Simple) Bayesian Network
  - Use the above model to compute the probability of any *conditional* event
  - Exercise: $P(x_1, x_2, y \mid x_3, x_4)$

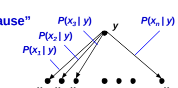
In-Class Exercise: Probabilistic Inference

- Inference Problem for a (Simple) Bayesian Network
  - Model: Naïve Bayes
  - Objective: compute the probability of any *conditional* event
- Exercise
  - Given
    - $P(x_i \mid y)$, $i \in \{1, 2, 3, 4\}$
    - $P(y)$
  - Want: $P(x_1, x_2, y \mid x_3, x_4)$

$$P(x_1, x_2, y \mid x_3, x_4) = \frac{P(x_3, x_4 \mid x_1, x_2, y)P(x_1, x_2, y)}{P(x_3, x_4)}$$
$$= \frac{P(x_1, x_2, x_3, x_4, y)}{P(x_3, x_4)}$$
$$= \frac{P(y)\prod_{i=1}^{4} P(x_i \mid y)}{\sum_{y} P(x_3 \mid y)P(x_4 \mid y)}$$

Unsupervised Learning and Conditional Independence

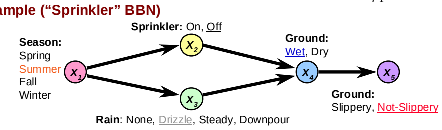
- Given: $(n + 1)$-Tuples $(x_1, x_2, \ldots, x_n, x_{n+1})$
  - No notion of instance variable or label
  - After seeing some examples, want to know something about the domain
    - Correlations among variables
    - Probability of certain events
    - Other properties
- Want to Learn: Most Likely Model 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.

Bayesian Belief Networks (BBNS): Definition

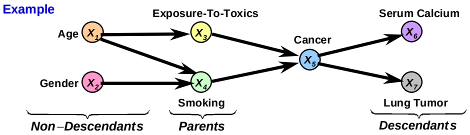
- **Conditional Independence**
  - *X* is conditionally independent (CI) from *Y* given *Z* (sometimes written $X \perp Y \mid Z$) iff $P(X \mid Y, Z) = P(X \mid Z)$ for all values of *X, Y,* and *Z*
  - Example: $P(Thunder \mid Rain, Lightning) = P(Thunder \mid Lightning) \Leftrightarrow T \perp R \mid L$
- **Bayesian Network**
  - Directed graph model of *conditional dependence assertions* (or *CI assumptions*)
  - Vertices (nodes): denote events (each a random variable)
  - Edges (arcs, links): denote conditional dependencies
- **General Product (Chain) Rule for BBNs** $P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid parents(X_i))$
- **Example ("Sprinkler" BBN)**



$P(Summer, Off, Drizzle, Wet, Not\text{-}Slippery) = P(S) \cdot P(O \mid S) \cdot P(D \mid S) \cdot P(W \mid O, D) \cdot P(N \mid W)$

Bayesian Belief Networks: Properties

- **Conditional Independence**
  - Variable (node): conditionally independent of non-descendants given parents
  - Example



  - Result: chain rule for probabilistic inference

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid Pa_i)$$

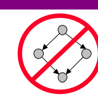
$$Pa_i = parents(X_i)$$

- **Bayesian Network: Probabilistic Semantics**
  - Node: variable
  - Edge: one axis of a conditional probability table (CPT)

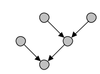
Bayesian Belief Networks: Inference

- **Problem Definition**
  - Given
    - Bayesian network with specified CPTs
    - Observed values for some nodes in network
  - Return: inferred (probabilities of) values for query node(s)
- **Implementation**
  - Bayesian network contains all information needed for this inference
    - If only one variable with unknown value, easy to infer it
    - In general case, problem is intractable ($\mathcal{NP}$-hard: reduction to *3-CNF-SAT*)
  - In practice, can succeed in many cases using different methods
    - Exact inference: work well for some network structures
    - Monte Carlo: "simulate" network to randomly calculate approximate solutions
  - Key machine learning issues
    - Feasible to elicit this information or learn it from data?
    - How to learn structure that makes inference more tractable?

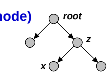
Tree Dependent Distributions



- **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(root)*, and *n* - 1 CPTs (1 per node)
  - All CPTs are 2-dimensional: *P(child | parent)*

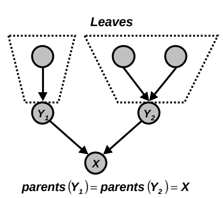


- **Independence Assumptions**
  - As for general BBN: *x* is independent of non-descendants given (single) parent *z*
  - *Very strong assumption* (applies in some domains but not most)

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

$$P(x) = P(X = x)$$
$$= \sum_{y_1, y_2} P(x \mid y_1, y_2) \cdot P(y_1, y_2)$$
$$= \sum_{y_1, y_2} P(x \mid y_1, y_2) \cdot P(y_1) \cdot P(y_2)$$

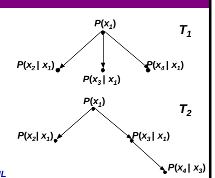


$parents(Y_1) = parents(Y_2) = X$

  - *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* ~ distribution $\mathcal{D}$
  - Return: *most likely tree T* that generated *D*
  - i.e., search for MAP hypothesis



- **MAP Estimation over BBNs**
  - Assuming uniform priors on trees, $h_{MAP} \equiv h_{ML} \equiv T_{ML}$

$$T_{ML} = \arg\max_{T \in H} P(D \mid T)$$

  - Maximization program

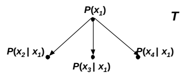
$$T_{ML} = \arg\max_{T \in H} \prod_{(x_1, x_2, \ldots, x_n) \in D} P_T(x_1, x_2, \ldots, x_n)$$
$$= \arg\max_{T \in H} \prod_{(x_1, x_2, \ldots, x_n) \in D} \prod_i P_T(x_i \mid parents(x_i))$$

  - Try this for Naïve Bayes…

In-Class Exercise: Learning Distributions [1]

- Input: $D \equiv \{1011, 1001, 0100\} \sim \mathcal{D}$
- 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

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

- CPT for P_3

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

- 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]

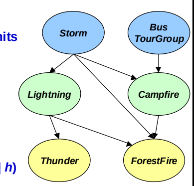
- Input Data: $D \equiv \{1011, 1001, 0100\}$
- Candidate Models: CPTs for Table (T_1), T_2, T_3
- Results: Likelihood Estimation
  - $P(D \mid T_1) = P(1011 \mid T_1) \cdot P(1011 \mid T_1) \cdot P(1011 \mid T_1) = 0.0 \cdot 0.0 \cdot 0.1 = \underline{0.0}$
  - $P(D \mid T_2) = P(1011 \mid T_2) \cdot P(1001 \mid T_2) \cdot P(0100 \mid T_2)$
    - $P(1011 \mid T_2) = P(x_4 = 1) \cdot P(x_1 = 1 \mid x_4 = 1) \cdot P(x_2 = 0 \mid x_4 = 1) \cdot P(x_e = 1 \mid x_4 = 1) = 1/2 \cdot 1/2 \cdot 1/3 \cdot 5/6 = 5/72$
    - $P(1001 \mid T_2) = 1/2 \cdot 1/2 \cdot 2/3 \cdot 5/6 = 10/72$
    - $P(0100 \mid T_2) = 1/2 \cdot 1/2 \cdot 2/3 \cdot 5/6 = 10/72$
    - $P(D \mid T_2) = 500 / 373248 \approx \underline{0.0013}$
  - $P(D \mid T_3) = \underline{1/27}$
  - 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, 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_1 (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 $P(D \mid h)$
- 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)

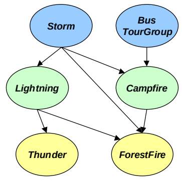
Learning Bayesian Networks: Gradient Ascent

- 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 $Y_i \equiv$ *Campfire*, then (for example) $u_{ik} \equiv \langle Storm = T, BusTourGroup = F\rangle$
  - WHILE termination condition not met DO // perform gradient ascent
    - Update all CPT entries w_{ijk} using training data D

$$w_{ijk} \leftarrow w_{ijk} + r \sum_{x \in D} \frac{P_h(y_{ij}, u_{ik} \mid x)}{w_{ijk}}$$

    - Renormalize w_{ijk} to assure invariants:

$$\sum_j w_{ijk} = 1$$
$$\forall j \,.\, 0 \le w_{ijk} \le 1$$



- 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:

$$P(T \mid D) \propto P(D \mid T) = \arg\max_{T \in H} \prod_{(x_1, x_2, \ldots, x_n) \in D} \prod_i P_T(x_i \mid parents(x_i))$$

  - RETURN the maximal T
- Is This Practical?
  - Typically not… (|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

$$D(P \,\|\, P') \equiv \sum_x P(x) \lg \frac{P(x)}{P'(x)}$$

  - 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. ~ *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

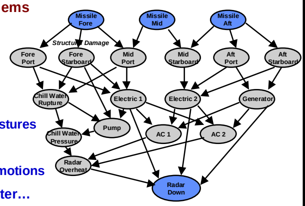
$$\mathrm{I}(X;Y) \equiv \sum_{x,y} P(x,y) \lg \frac{P(x,y)}{P(x) \cdot P(y)} = \mathrm{D}(P(X,Y) \,\|\, P(X) \cdot P(Y))$$

  - 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 Voloshin, 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

Terminology

- Graphical Models of Probability
  - Bayesian belief networks (BBNs) *aka* belief networks *aka* causal networks
  - Conditional independence, causal Markovity
  - Inference and learning using Bayesian networks
    - Representation of distributions: conditional probability tables (CPTs)
    - Learning polytrees (singly-connected BBNs) and tree-structured BBNs (trees)
- BBN Inference
  - Type of probabilistic reasoning
  - Finds answer to query about *P*(*x*) - *aka* QA
- Gradient Learning in BBNs
  - Known structure
  - Partial observability
- Structure Learning for Trees
  - Kullback-Leibler distance (K-L divergence, cross-entropy, relative entropy)
  - Maximum weight spanning tree (MWST) algorithm

Summary Points

- Graphical Models of Probability
  - Bayesian networks: introduction
    - Definition and basic principles
    - Conditional independence (causal Markovity) assumptions, tradeoffs
  - Inference and learning using Bayesian networks
    - Acquiring and applying CPTs
    - Searching the space of trees: max likelihood
    - Examples: *Sprinkler*, *Cancer*, *Forest-Fire*, generic tree learning
- CPT Learning: Gradient Algorithm *Train-BN*
- Structure Learning in Trees: MWST Algorithm *Learn-Tree-Structure*
- Reasoning under Uncertainty: Applications and Augmented Models
- Some Material From: http://robotics.Stanford.EDU/~koller
- Next Lecture: Read Heckerman Tutorial