

CSCE 478/878 Lecture 6: Bayesian Learning

Stephen D. Scott
(Adapted from Tom Mitchell's slides)

October 27, 2008

Bayesian Methods

Not all hypotheses are created equal (even if they are all consistent with the training data)

Might have reasons (domain information) to favor some hypotheses over others *a priori*

Bayesian methods work with probabilities, and have two main roles:

1. Provide practical learning algorithms:

- Naïve Bayes learning
- Bayesian belief network learning
- Combine prior knowledge (prior probabilities) with observed data
- Requires prior probabilities

2. Provides useful conceptual framework

- Provides “gold standard” for evaluating other learning algorithms
- Additional insight into Occam’s razor

Outline

- Bayes Theorem
- MAP, ML hypotheses
- MAP learners
- Minimum description length principle
- Bayes optimal classifier/Gibbs algorithm
- Naïve Bayes classifier
- Bayesian belief networks

Bayes Theorem

In general, an identity for conditional probabilities

For our work, we want to know the probability that a particular $h \in H$ is the correct hypothesis given that we have seen training data D (examples and labels). Bayes theorem lets us do this.

$$P(h | D) = \frac{P(D | h)P(h)}{P(D)}$$

- $P(h)$ = prior probability of hypothesis h (might include domain information)
- $P(D)$ = probability of training data D
- $P(h | D)$ = probability of h given D
- $P(D | h)$ = probability of D given h

Note $P(h | D)$ increases with $P(D | h)$ and $P(h)$ and decreases with $P(D)$

Choosing Hypotheses

$$P(h | D) = \frac{P(D | h)P(h)}{P(D)}$$

Generally want the most probable hypothesis given the training data

Maximum a posteriori hypothesis h_{MAP} :

$$\begin{aligned} h_{MAP} &= \operatorname{argmax}_{h \in H} P(h | D) \\ &= \operatorname{argmax}_{h \in H} \frac{P(D | h)P(h)}{P(D)} \\ &= \operatorname{argmax}_{h \in H} P(D | h)P(h) \end{aligned}$$

If assume $P(h_i) = P(h_j)$ for all i, j , then can further simplify, and choose the *maximum likelihood* (ML) hypothesis

$$h_{ML} = \operatorname{argmax}_{h_i \in H} P(D | h_i)$$

Bayes Theorem Example

Does patient have cancer or not?

A patient takes a lab test and the result comes back positive. The test returns a correct positive result in only 98% of the cases in which the disease is actually present, and a correct negative result in only 97% of the cases in which the disease is not present. Furthermore, .008 of the entire population have this cancer.

$$\begin{array}{ll} P(\text{cancer}) = & P(\neg\text{cancer}) = \\ P(+ | \text{cancer}) = & P(- | \text{cancer}) = \\ P(+ | \neg\text{cancer}) = & P(- | \neg\text{cancer}) = \end{array}$$

Now consider new patient for whom the test is positive. What is our diagnosis?

$$\begin{array}{l} P(+ | \text{cancer})P(\text{cancer}) = \\ P(+ | \neg\text{cancer})P(\neg\text{cancer}) = \end{array}$$

$$\text{So } h_{MAP} =$$

Basic Formulas for Probabilities

- *Product Rule*: probability $P(A \wedge B)$ of a conjunction of two events A and B:

$$P(A \wedge B) = P(A | B)P(B) = P(B | A)P(A)$$

- *Sum Rule*: probability of a disjunction of two events A and B:

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$

- *Theorem of total probability*: if events A_1, \dots, A_n are mutually exclusive with $\sum_{i=1}^n P(A_i) = 1$, then

$$P(B) = \sum_{i=1}^n P(B | A_i)P(A_i)$$

Brute Force MAP Hypothesis Learner

1. For each hypothesis h in H , calculate the posterior probability

$$P(h | D) = \frac{P(D | h)P(h)}{P(D)}$$

2. Output the hypothesis h_{MAP} with the highest posterior probability

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

Problem: what if H exponentially or infinitely large?

Relation to Concept Learning

Consider our usual concept learning task: instance space X , hypothesis space H , training examples D

Consider the Find-S learning algorithm (outputs most specific hypothesis from the version space $VS_{H,D}$)

What would brute-force MAP learner output as MAP hypothesis?

Does Find-S output a MAP hypothesis??

Relation to Concept Learning (cont'd)

Assume fixed set of instances $\langle x_1, \dots, x_m \rangle$

Assume D is the set of classifications

$$D = \langle c(x_1), \dots, c(x_m) \rangle$$

Assume no noise and $c \in H$, so choose

$$P(D | h) = \begin{cases} 1 & \text{if } d_i = h(x_i) \text{ for all } d_i \in D \\ 0 & \text{otherwise} \end{cases}$$

Choose $P(h) = 1/|H| \forall h \in H$, i.e. uniform dist.

If h inconsistent with D , then

$$P(h | D) = (0 \cdot P(h)) / P(D) = 0$$

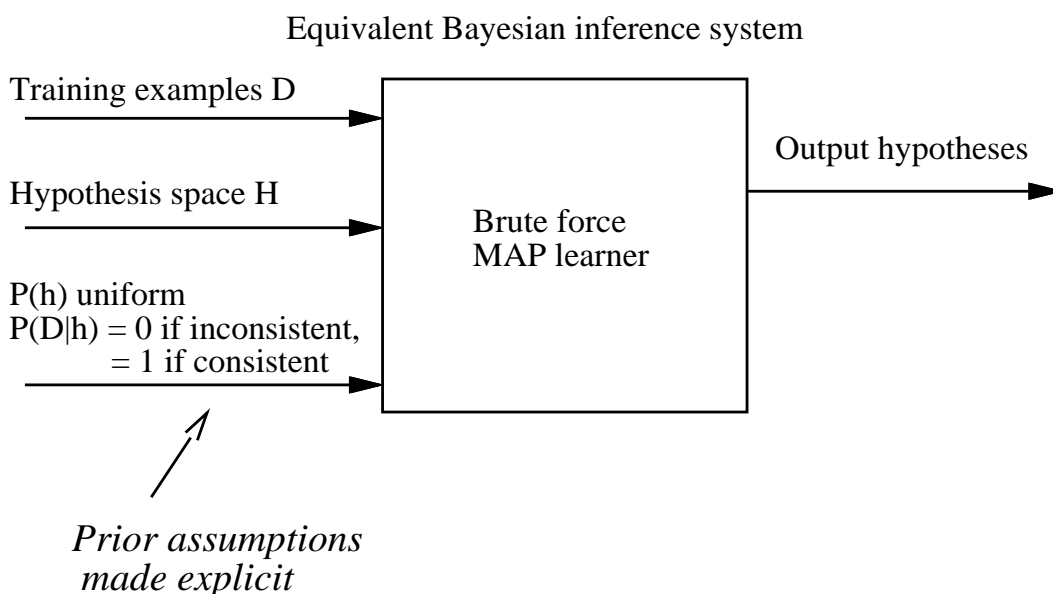
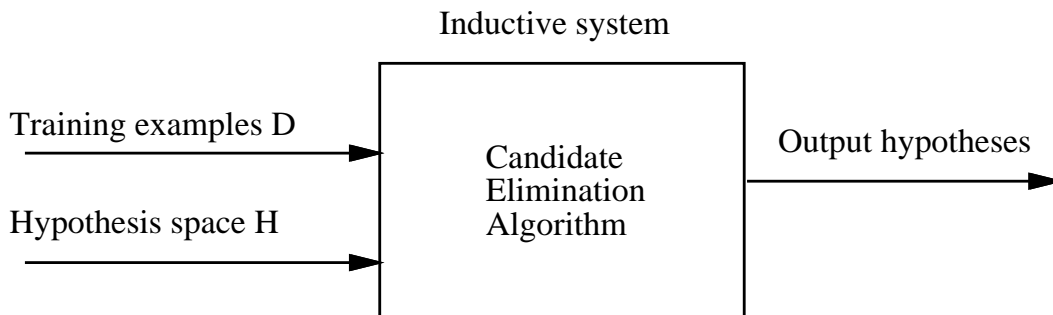
If h consistent with D , then

$$\begin{aligned} P(h | D) &= (1 \cdot 1/|H|) / P(D) = (1/|H|) / (|VS_{H,D}|/|H|) \\ &= 1/|VS_{H,D}| \text{ (see Thrm of total prob., slide 7)} \end{aligned}$$

Thus if D noise-free and $c \in H$ and $P(h)$ uniform,

every consistent hypothesis is a MAP hypothesis

Characterizing Learning Algorithms by Equivalent MAP Learners



So we can characterize algorithms in a Bayesian framework even though they don't directly manipulate probabilities

Other priors will allow Find-S, etc. to output MAP; e.g. $P(h)$ that favors more specific hypotheses

Learning A Real-Valued Function

Consider any real-valued target function f

Training examples $\langle x_i, d_i \rangle$, where d_i is noisy training value

- $d_i = f(x_i) + e_i$
- e_i is random variable (noise) drawn independently for each x_i according to some Gaussian distribution with mean $\mu_{e_i} = 0$

Then the maximum likelihood hypothesis h_{ML} is the one that minimizes the sum of squared errors, e.g. a linear unit trained with GD/EG:

$$h_{ML} = \operatorname{argmin}_{h \in H} \sum_{i=1}^m (d_i - h(x_i))^2$$

Learning A Real-Valued Function (cont'd)

$$\begin{aligned}h_{ML} &= \operatorname{argmax}_{h \in H} p(D | h) = \operatorname{argmax}_{h \in H} p(d_1, \dots, d_m | h) \\ &= \operatorname{argmax}_{h \in H} \prod_{i=1}^m p(d_i | h) \quad (\text{if } d_i\text{'s cond. indep.}) \\ &= \operatorname{argmax}_{h \in H} \prod_{i=1}^m \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2} \left(\frac{d_i - h(x_i)}{\sigma}\right)^2\right) \\ (\mu_{e_i} = 0 &\Rightarrow \mathbf{E}[d_i | h] = h(x_i))\end{aligned}$$

Maximize natural log instead:

$$\begin{aligned}h_{ML} &= \operatorname{argmax}_{h \in H} \sum_{i=1}^m \ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2} \left(\frac{d_i - h(x_i)}{\sigma}\right)^2 \\ &= \operatorname{argmax}_{h \in H} \sum_{i=1}^m -\frac{1}{2} \left(\frac{d_i - h(x_i)}{\sigma}\right)^2 \\ &= \operatorname{argmax}_{h \in H} \sum_{i=1}^m -(d_i - h(x_i))^2 \\ &= \operatorname{argmin}_{h \in H} \sum_{i=1}^m (d_i - h(x_i))^2\end{aligned}$$

Thus have Bayesian justification for minimizing squared error (under certain assumptions)

Learning to Predict Probabilities

Consider predicting survival probability from patient data

Training examples $\langle x_i, d_i \rangle$, where d_i is 1 or 0
(assume label is [or appears] probabilistically generated)

Want to train neural network to output the probability that x_i has label 1, not the label itself

Using approach similar to previous slide (p. 169), can show

$$h_{ML} = \operatorname{argmax}_{h \in H} \sum_{i=1}^m d_i \ln h(x_i) + (1 - d_i) \ln(1 - h(x_i))$$

i.e. find h minimizing cross-entropy

For single sigmoid unit, use update rule

$$w_j \leftarrow w_j + \eta \sum_{i=1}^m (d_i - h(x_i)) x_{ij}$$

to find h_{ML} (can also derive EG rule)

Minimum Description Length Principle

Occam's razor: prefer the shortest hypothesis

MDL: prefer the hypothesis h that satisfies

$$h_{MDL} = \operatorname{argmin}_{h \in H} L_{C_1}(h) + L_{C_2}(D | h)$$

where $L_C(x)$ is the description length of x under encoding C

Example: H = decision trees, D = training data labels

- $L_{C_1}(h)$ is # bits to describe tree h
- $L_{C_2}(D | h)$ is # bits to describe D given h
 - Note $L_{C_2}(D | h) = 0$ if examples classified perfectly by h . Need only describe exceptions
- Hence h_{MDL} trades off tree size for training errors

Minimum Description Length Principle

Bayesian Justification

$$\begin{aligned}h_{MAP} &= \operatorname{argmax}_{h \in H} P(D | h)P(h) \\ &= \operatorname{argmax}_{h \in H} \log_2 P(D | h) + \log_2 P(h) \\ &= \operatorname{argmin}_{h \in H} -\log_2 P(D | h) - \log_2 P(h) \quad (1)\end{aligned}$$

Interesting fact from information theory: The optimal (shortest expected coding length) code for an event with probability p is $-\log_2 p$ bits.

So interpret (1):

- $-\log_2 P(h)$ is length of h under optimal code
- $-\log_2 P(D | h)$ is length of D given h under optimal code

→ prefer the hypothesis that minimizes

$$\text{length}(h) + \text{length}(\text{misclassifications})$$

Caveat: $h_{MDL} = h_{MAP}$ doesn't apply for arbitrary encodings (need $P(h)$ and $P(D | h)$ to be optimal); merely a guide

Bayes Optimal Classifier

- So far we've sought the most probable hypothesis given the data D , i.e. h_{MAP}
- But given new instance x , $h_{MAP}(x)$ is not necessarily the most probable classification!

- Consider three possible hypotheses:

$$P(h_1 | D) = 0.4, P(h_2 | D) = 0.3, P(h_3 | D) = 0.3$$

Given new instance x ,

$$h_1(x) = +, h_2(x) = -, h_3(x) = -$$

- $h_{MAP}(x) =$
- What's the most probable classification of x ?

Bayes Optimal Classifier (cont'd)

Bayes optimal classification:

$$\operatorname{argmax}_{v_j \in V} \sum_{h_i \in H} P(v_j | h_i) P(h_i | D)$$

where V is set of possible labels (e.g. $\{+, -\}$)

Example:

$$P(h_1 | D) = 0.4, \quad P(- | h_1) = 0, \quad P(+ | h_1) = 1$$

$$P(h_2 | D) = 0.3, \quad P(- | h_2) = 1, \quad P(+ | h_2) = 0$$

$$P(h_3 | D) = 0.3, \quad P(- | h_3) = 1, \quad P(+ | h_3) = 0$$

therefore

$$\sum_{h_i \in H} P(+ | h_i) P(h_i | D) = 0.4$$

$$\sum_{h_i \in H} P(- | h_i) P(h_i | D) = 0.6$$

and

$$\operatorname{argmax}_{v_j \in V} \sum_{h_i \in H} P(v_j | h_i) P(h_i | D) = -$$

On average, no other classifier using same prior and same hyp. space can outperform Bayes optimal!

Gibbs Algorithm

Bayes optimal classifier provides best result, but can be expensive or impossible if many hypotheses [Though some cases can be made efficient, if one assumes particular probability distributions.]

Gibbs algorithm:

1. Randomly choose one hypothesis according to $P(h | D)$
2. Use this to classify new instance

Surprising fact: Assume target concepts are drawn at random from H according to priors on H . Then:

$$E [error_{Gibbs}] \leq 2 E [error_{Bayes\ Optimal}]$$

i.e. if prior correct and $c \in H$, then average error at most twice best possible!

E.g. Suppose correct, uniform prior distribution over H . Then

- Pick any hypothesis from V_S with uniform probability
- Expected error no worse than twice Bayes optimal

Still have to be able to choose random hypothesis!

Naïve Bayes Classifier

Along with decision trees, neural networks, nearest neighbor, SVMs, boosting, one of the most practical learning methods

When to use

- Moderate or large training set available
- Attributes that describe instances are conditionally independent given classification

Successful applications:

- Diagnosis
- Classifying text documents

Naïve Bayes Classifier (cont'd)

Assume target function $f : X \rightarrow V$, where each instance x described by attributes $\langle a_1, a_2, \dots, a_n \rangle$

Most probable value of $f(x)$ is:

$$\begin{aligned} v_{MAP} &= \operatorname{argmax}_{v_j \in V} P(v_j \mid a_1, a_2, \dots, a_n) \\ &= \operatorname{argmax}_{v_j \in V} \frac{P(a_1, a_2, \dots, a_n \mid v_j) P(v_j)}{P(a_1, a_2, \dots, a_n)} \\ &= \operatorname{argmax}_{v_j \in V} P(a_1, a_2, \dots, a_n \mid v_j) P(v_j) \end{aligned}$$

Problem with estimating probs from training data: estimating $P(v_j)$ easily done by counting, but there are exponentially (in n) many combs. of values of a_1, \dots, a_n , so can't get estimates for most combs

Naïve Bayes assumption:

$$P(a_1, a_2, \dots, a_n \mid v_j) = \prod_i P(a_i \mid v_j)$$

so naïve Bayes classifier:

$$v_{NB} = \operatorname{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i \mid v_j)$$

Now have only polynomial number of probs to estimate

Naïve Bayes Algorithm

Naïve_Bayes_Learn

1. For each target value v_j

(a) $\hat{P}(v_j) \leftarrow$ estimate $P(v_j)$ = fraction of exs with v_j

(b) For each attribute value a_i of each attrib a

i. $\hat{P}(a_i | v_j) \leftarrow$ estimate $P(a_i | v_j)$ = fraction of v_j -labeled exs with a_i

Classify_New_Instance(x)

$$v_{NB} = \operatorname{argmax}_{v_j \in V} \hat{P}(v_j) \prod_{a_i \in x} \hat{P}(a_i | v_j)$$

Naïve Bayes Example

Training Examples (Table 3.2):

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Example to classify:

$\langle \text{Outlk} = \text{sun}, \text{Temp} = \text{cool}, \text{Humid} = \text{high}, \text{Wind} = \text{strong} \rangle$

Assign label $v_{NB} = \operatorname{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$

$$P(y) \cdot P(\text{sun} | y) \cdot P(\text{cool} | y) \cdot P(\text{high} | y) \cdot P(\text{strong} | y)$$

$$= (9/14) \cdot (2/9) \cdot (3/9) \cdot (3/9) \cdot (3/9) = 0.0053$$

$$P(n) P(\text{sun} | n) P(\text{cool} | n) P(\text{high} | n) P(\text{strong} | n)$$

$$= (5/14) \cdot (3/5) \cdot (1/5) \cdot (4/5) \cdot (3/5) = 0.0206$$

So $v_{NB} = n$

Naïve Bayes Subtleties

- Conditional independence assumption is often violated, i.e.

$$P(a_1, a_2, \dots, a_n | v_j) \neq \prod_i P(a_i | v_j)$$

... but it works surprisingly well anyway. Note don't need estimated posteriors $\hat{P}(v_j | x)$ to be correct; need only that

$$\operatorname{argmax}_{v_j \in V} \hat{P}(v_j) \prod_i \hat{P}(a_i | v_j) = \operatorname{argmax}_{v_j \in V} P(v_j) P(a_1, \dots, a_n | v_j)$$

Sufficient conditions given in
[Domingos & Pazzani, 1996]

Naïve Bayes

Subtleties (cont'd)

- What if none of the training instances with target value v_j have attribute value a_i ? Then

$$\hat{P}(a_i | v_j) = 0, \text{ and } \hat{P}(v_j) \prod_i \hat{P}(a_i | v_j) = 0$$

Typical solution is to use as estimate:

$$\hat{P}(a_i | v_j) \leftarrow \frac{n_c + mp}{n + m}$$

where

- n is number of training examples for which $v = v_j$,
- n_c number of examples for which $v = v_j$ and $a = a_i$
- p is prior estimate for $\hat{P}(a_i | v_j)$
- m is weight given to prior (i.e. number of “virtual” examples)
- Sometimes called “pseudocounts”

Naïve Bayes

Application: Learning to Classify Text

- Target concept *Interesting?* : $Document \rightarrow \{+, -\}$
(can you also use NB as a ranker?)
- Each document is a vector of words (i.e. one attribute per word position), e.g. $a_1 = \text{"our"}$, $a_2 = \text{"approach"}$, etc.
- Naïve Bayes very effective despite obvious violation of conditional independence assumption
- See Section 6.10 for more detail

Bayesian Belief Networks

- Sometimes naïve Bayes assumption of conditional independence too restrictive
- But inferring probabilities is intractable without some such assumptions
- Bayesian belief networks (also called Bayes Nets) describe conditional independence among subsets of variables
- Allows combining prior knowledge about dependencies among variables with observed training data

Conditional Independence

Definition: X is conditionally independent of Y given Z if the probability distribution governing X is independent of the value of Y given the value of Z ; that is, if

$$(\forall x_i, y_j, z_k) P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z = z_k)$$

more compactly, we write

$$P(X | Y, Z) = P(X | Z)$$

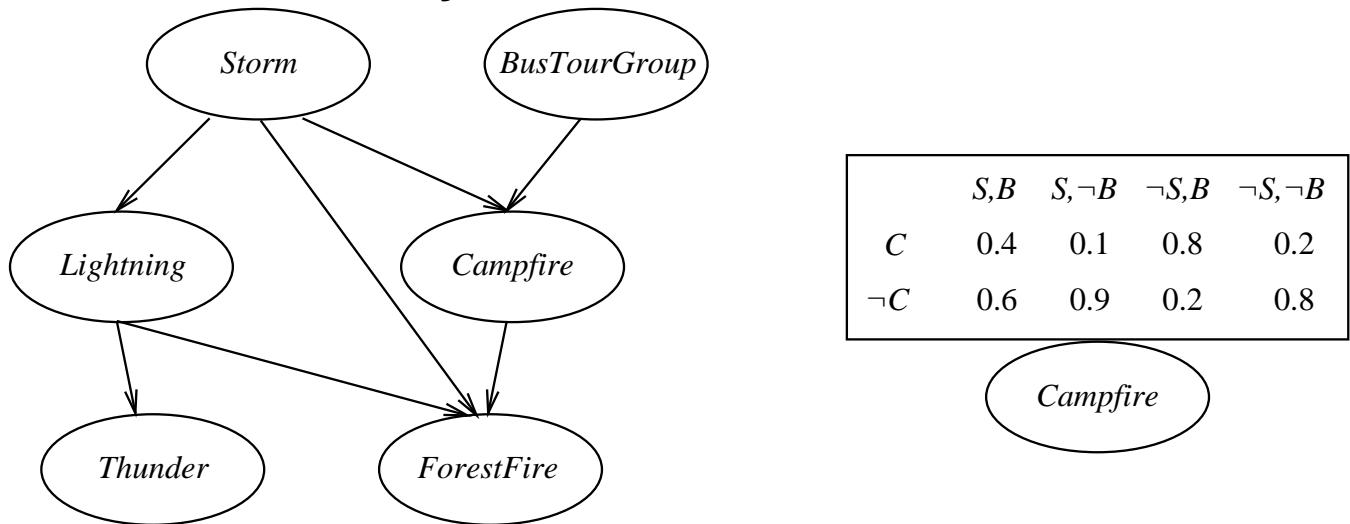
Example: *Thunder* is conditionally independent of *Rain*, given *Lightning*

$$P(\textit{Thunder} | \textit{Rain}, \textit{Lightning}) = P(\textit{Thunder} | \textit{Lightning})$$

Naïve Bayes uses conditional independence and product rule (slide 7) to justify

$$\begin{aligned} P(X, Y | Z) &= P(X | Y, Z)P(Y | Z) \\ &= P(X | Z)P(Y | Z) \end{aligned}$$

Bayesian Belief Network

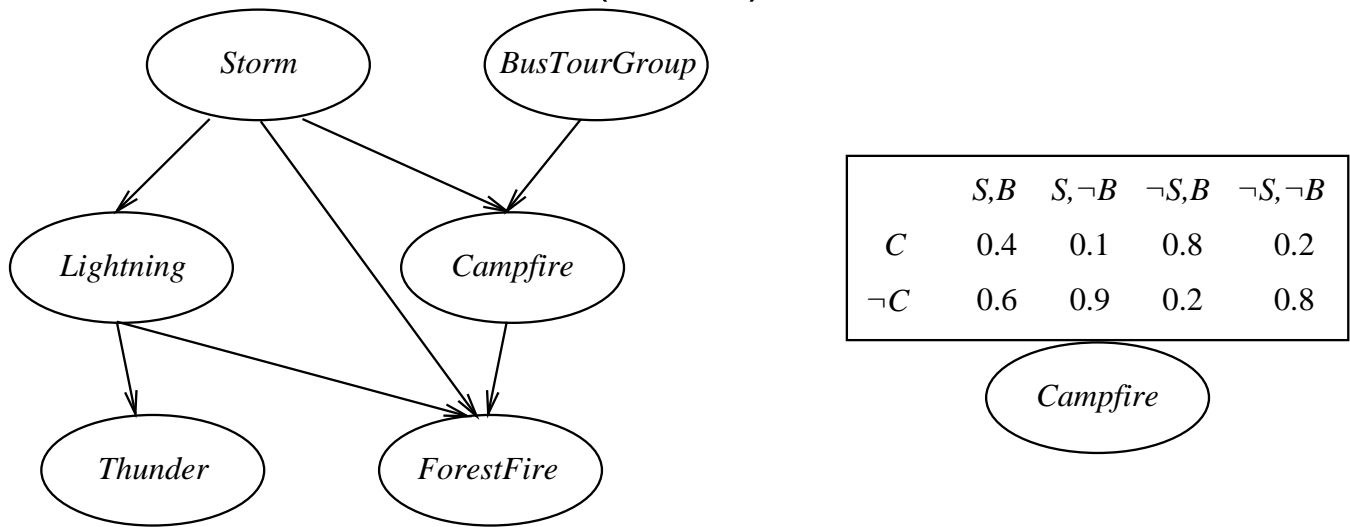


Network represents a set of conditional independence assertions:

- Each node is asserted to be conditionally independent of its nondescendants, given its immediate predecessors
- Directed acyclic graph

Bayesian Belief Network

(cont'd)



Represents joint probability distribution over all network variables $\langle Y_1, \dots, Y_n \rangle$, e.g.

$$P(\text{Storm}, \text{BusTourGroup}, \dots, \text{ForestFire})$$

- In general, for $y_i = \text{value of } Y_i$

$$P(y_1, \dots, y_n) = \prod_{i=1}^n P(y_i \mid \text{Parents}(Y_i))$$

where $\text{Parents}(Y_i)$ denotes immediate predecessors of Y_i in graph

- E.g. $P(S, B, C, \neg L, \neg T, \neg F) =$

$$P(S) \cdot P(B) \cdot \underbrace{P(C \mid B, S)}_{0.4} \cdot P(\neg L \mid S) \cdot P(\neg T \mid \neg L) \cdot P(\neg F \mid S, \neg L, \neg C)$$

Inference in Bayesian Networks

Want to infer the probabilities of values of one or more network variables (attributes), given observed values of others, i.e. want the probability distribution of a subset of variables given the values of a subset of the other variables

- Bayes net contains all information needed for this inference: can simply brute force try all combinations of values of the unknown variables
- Of course, this takes time exponential in number of unknowns
- In general case, problem is NP hard

In practice, can succeed in many cases

- Exact inference methods work well for some network structures
- Monte Carlo methods “simulate” the network randomly to calculate approximate solutions

Learning of Bayesian Networks

We know how to use Bayesian Networks, but how do we learn one?

Several variants of this learning task

- Network structure might be known or unknown
- Training examples might provide values of all network variables, or just some

If structure known and all variables observed, then it's as easy as training a naïve Bayes classifier (just count occurrences as before)

Learning of Bayesian Networks

(cont'd)

Suppose structure known, variables partially observable

E.g. observe *ForestFire*, *Storm*, *BusTourGroup*, *Thunder*,
but not *Lightning*, *Campfire*

- Similar to training neural network with hidden units; in fact can learn network conditional probability tables using gradient ascent
- Converge to network h that (locally) maximizes $P(D | h)$, i.e. search for ML hypothesis
- Can also use EM (expectation maximization) algorithm
 - Use observations of variables to predict their values in cases when they're not observed
 - EM has many other applications, e.g. hidden Markov models (HMMs) used for e.g. biological sequence analysis

Bayesian Belief Networks

Summary

- Combine prior knowledge with observed data
- Impact of prior knowledge (when correct!) is to lower the sample complexity
- Active research area
 - Extend from boolean to real-valued variables
 - Parameterized distributions instead of tables
 - More effective inference methods
- Will cover in much more depth in CSCE 970 in Spring

Topic summary due in 1 week!