

4 Bayesian Belief Networks

(also called Bayes Nets)

Interesting because:

- The Naive Bayes assumption of conditional independence of attributes is too restrictive.
(But it's intractable without some such assumptions...)
- Bayesian Belief networks describe **conditional independence among *subsets* of variables**.
- It allows the combination of prior knowledge about (in)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 a value of Z :

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

More compactly, we write $P(X|Y, Z) = P(X|Z)$

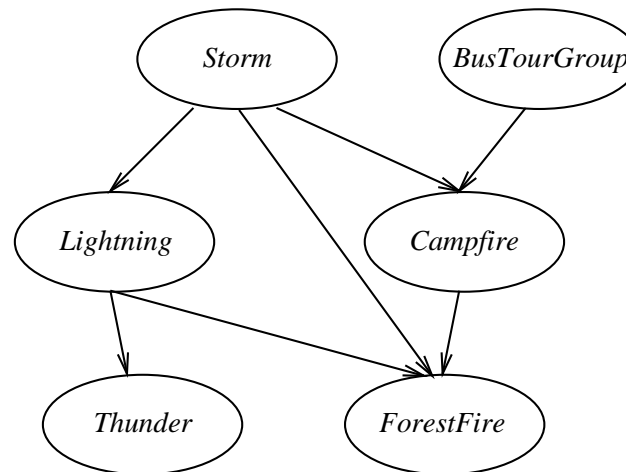
Note: Naive Bayes uses conditional independence to justify

$$P(A_1, A_2 | V) = P(A_1 | A_2, V) P(A_2 | V) = P(A_1 | V) P(A_2 | V)$$

Generalizing the above definition:

$$P(X_1 \dots X_l | Y_1 \dots Y_m, Z_1 \dots Z_n) = P(X_1 \dots X_l | Z_1 \dots Z_n)$$

A Bayes Net



	S, B	$S, \neg B$	$\neg S, B$	$\neg S, \neg B$
C	0.4	0.1	0.8	0.2
$\neg C$	0.6	0.9	0.2	0.8



The network is defined by

- A directed acyclic graph, representing a set of conditional independence assertions:

Each node — representing a random variable — is asserted to be conditionally independent of its nondescendants, given its immediate predecessors.

Example: $P(\text{Thunder} | \text{ForestFire}, \text{Lightning}) = P(\text{Thunder} | \text{Lightning})$

- A table of local conditional probabilities for each node/variable.

A Bayes Net (Cont'd)

represents **the joint probability distribution** over all variables Y_1, Y_2, \dots, Y_n :

This joint distribution is fully defined by the graph, plus the conditional probabilities:

$$P(y_1, \dots, y_n) = P(Y_1 = y_1, \dots, Y_n = y_n) = \prod_{i=1}^n P(y_i | Parents(Y_i))$$

where $Parents(Y_i)$ denotes immediate predecessors of Y_i in the graph.

In our **example**: $P(Storm, BusTourGroup, \dots, ForestFire)$

Inference in Bayesian Nets

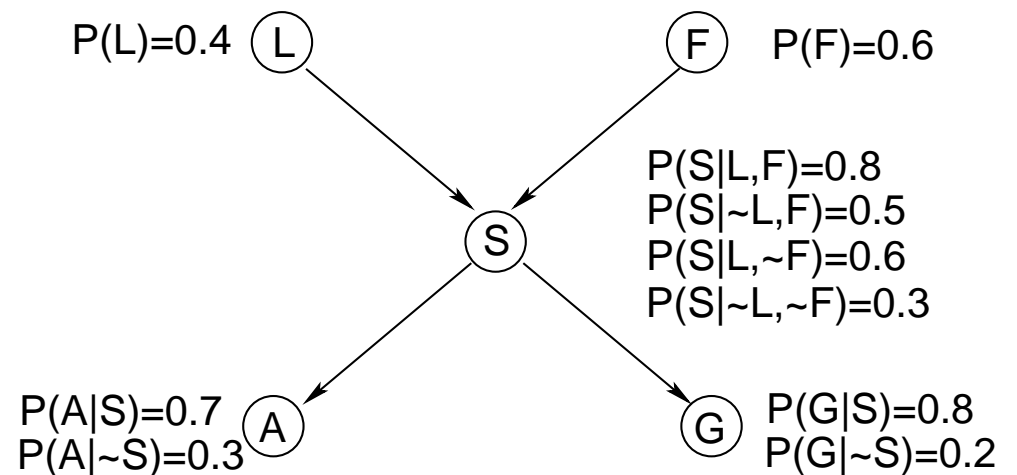
Question: Given a Bayes net, can one infer the probabilities of values of one or more network variables, given the observed values of (some) others?

Example:

Given the Bayes net

compute:

- (a) $P(S)$
- (b) $P(A, S)$
- (b) $P(A)$



Inference in Bayesian Nets (Cont'd)

Answer(s):

- If only one variable is of unknown (probability) value, then it is easy to infer it
- In the general case, we can compute the probability distribution for any subset of network variables, given the distribution for any subset of the remaining variables.
But...
- The exact inference of probabilities for an arbitrary Bayes net is an NP-hard problem!!

Inference in Bayesian Nets (Cont'd)

In practice, we can succeed in many cases:

- Exact inference methods work well for some net structures.
- Monte Carlo methods “simulate” the network randomly to calculate approximate solutions [Pradham & Dagum, 1996].

(In theory even approximate inference of probabilities in Bayes Nets can be NP-hard!! [Dagum & Luby, 1993])

Learning Bayes Nets (I)

There are **several variants** of this learning task

- The network structure might be either *known* or *unknown* (i.e., it has to be inferred from the training data).
- The training examples might provide values of *all* network variables, or just for *some* of them.

The simplest case:

If the structure is known and we can observe the values of all variables,

then it is easy to estimate the conditional probability table entries. (Analogous to training a Naive Bayes classifier.)

Learning Bayes Nets (II)

When

- the structure of the Bayes Net is known, and
- the variables are only partially observable in the training data

learning the entries in the conditional probabilities tables is similar to (learning the weights of hidden units in) training a neural network with hidden units:

- We can learn the net's conditional probability tables using the gradient ascent!
- Converge to the network h that (locally) maximizes $P(D|h)$.

Gradient Ascent for Bayes Nets

49.

Let w_{ijk} denote one entry in the conditional probability table for the variable Y_i in the network

$$w_{ijk} = P(Y_i = y_{ij} | Parents(Y_i) = \text{the list } u_{ik} \text{ of values})$$

It can be shown (see the next two slides) that

$$\frac{\partial \ln P_h(D)}{\partial w_{ijk}} = \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik} | d)}{w_{ijk}}$$

therefore perform gradient ascent by repeatedly

1. **update all** w_{ijk} using the training data D
2. **renormalize the** w_{ijk} to assure

$$w_{ijk} \leftarrow w_{ijk} + \eta \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik} | d)}{w_{ijk}}$$

$$\sum_j w_{ijk} = 1 \text{ and } 0 \leq w_{ijk} \leq 1$$

Gradient Ascent for Bayes Nets: Calculus

$$\frac{\partial \ln P_h(D)}{\partial w_{ijk}} = \frac{\partial}{\partial w_{ijk}} \ln \prod_{d \in D} P_h(d) = \sum_{d \in D} \frac{\partial \ln P_h(d)}{\partial w_{ijk}} = \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial P_h(d)}{\partial w_{ijk}}$$

Summing over all values $y_{ij'}$ of Y_i , and $u_{ik'}$ of $U_i = \text{Parents}(Y_i)$:

$$\begin{aligned} \frac{\partial \ln P_h(D)}{\partial w_{ijk}} &= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} \sum_{j'k'} P_h(d|y_{ij'}, u_{ik'}) P_h(y_{ij'}, u_{ik'}) \\ &= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} \sum_{j'k'} P_h(d|y_{ij'}, u_{ik'}) P_h(y_{ij'}|u_{ik'}) P_h(u_{ik'}) \end{aligned}$$

Note that $w_{ijk} \equiv P_h(y_{ij}|u_{ik})$, therefore...

Gradient Ascent for Bayes Nets: Calculus (Cont'd)

$$\begin{aligned}
 \frac{\partial \ln P_h(D)}{\partial w_{ijk}} &= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} P_h(d|y_{ij}, u_{ik}) w_{ijk} P_h(u_{ik}) \\
 &= \sum_{d \in D} \frac{1}{P_h(d)} P_h(d|y_{ij}, u_{ik}) P_h(u_{ik}) \quad (\text{applying Bayes th.}) \\
 &= \sum_{d \in D} \frac{1}{P_h(d)} \frac{P_h(y_{ij}, u_{ik}|d) P_h(d) P_h(u_{ik})}{P_h(y_{ij}, u_{ik})} \\
 &= \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik}|d) P_h(u_{ik})}{P_h(y_{ij}, u_{ik})} = \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik}|d)}{P_h(y_{ij}|u_{ik})} \\
 &= \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik}|d)}{w_{ijk}}
 \end{aligned}$$

Learning Bayes Nets (II, Cont'd)

The **EM** algorithm (see next slides) can also be used.

Repeatedly:

1. Calculate/estimate from data the probabilities of unobserved variables w_{ijk} ,
assuming that the hypothesis h holds
2. Calculate a new h (i.e. new values of w_{ijk}) so to maximize $E[\ln P(D|h)]$,
where D now includes both the observed and the unobserved variables.

Learning Bayes Nets (III)

When the **structure is unknown**, algorithms usually use greedy search to trade off network complexity (add/subtract edges/nodes) against degree of fit to the data.

Example: [Cooper & Herscovitz, 1992] the *K2* algorithm:

When data is fully observable, use a score metric to choose among alternative networks.

They report an experiment on (re-learning) a network with 37 nodes and 46 arcs describing anesthesia problems in a hospital operating room. Using 3000 examples, the program succeeds almost perfectly: it misses one arc and adds an arc which is not in the original net.

Summary: Bayesian Belief Networks

- Combine prior knowledge with observed data
- The impact of prior knowledge (when correct!) is to lower the sample complexity
- Active/Recent research area
 - Extend from boolean to real-valued variables
 - Parameterized distributions instead of tables
 - Extend to first-order instead of propositional systems
 - More effective inference methods
 - ...