Bayesian Learning

Preview

- Bayes' theorem
- MAP learners
- Bayes optimal classifier
- Naive Bayes learner
- Example: text classification
- Bayesian networks
- EM algorithm

Two Roles for Bayesian Methods

Practical learning algorithms:

- Naive Bayes learning
- Bayesian network learning
- Combine prior knowledge with observed data
- Require prior probabilities

Useful conceptual framework:

- "Gold standard" for evaluating other learners
- Tools for analysis

Bayes' Theorem

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

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

Choosing Hypotheses

Find most probable hypothesis given training data

Maximum a posteriori hypothesis h_{MAP}:

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

Assuming $P(h_i) = P(h_j)$ we can further simplify, and choose the *Maximum likelihood* (ML) hypothesis

$$h_{ML} = \arg\max_{h_i \in H} P(D|h_i)$$

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, 0.008 of the entire population have this cancer.

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

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 $A_1, \ldots, A_n$ 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

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

Relation to Concept Learning

Let $D = \langle c(x_1), \ldots, c(x_m) \rangle$ (examples' classes)

Choose $P(D|h)$

- $P(D|h) = 1$ if h consistent with D
- $P(D|h) = 0$ otherwise

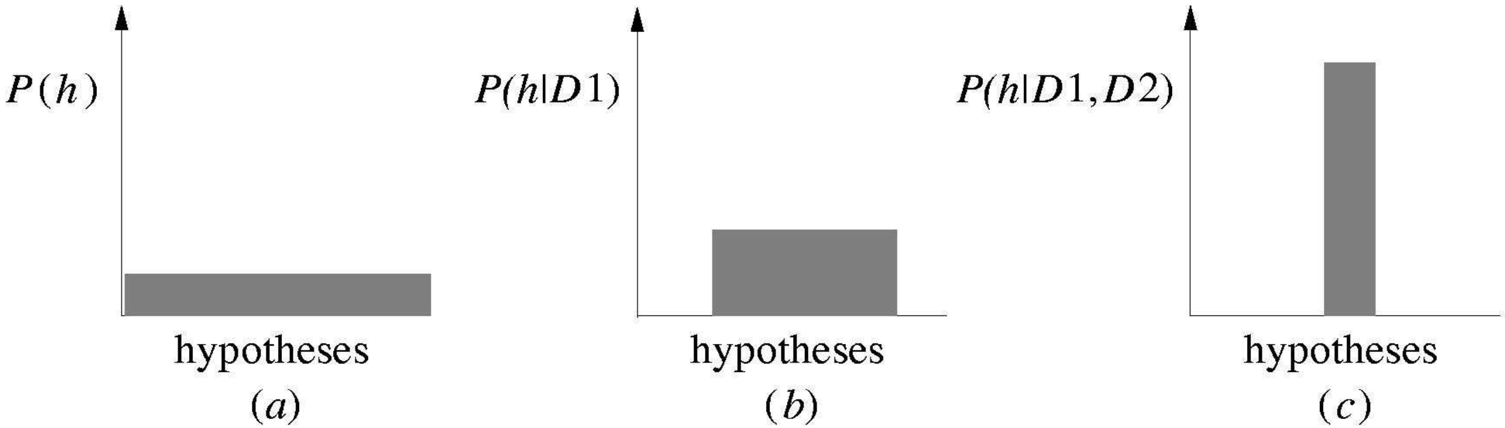
Choose $P(h)$ to be *uniform* distribution

- $P(h) = \frac{1}{|H|}$ for all h in H

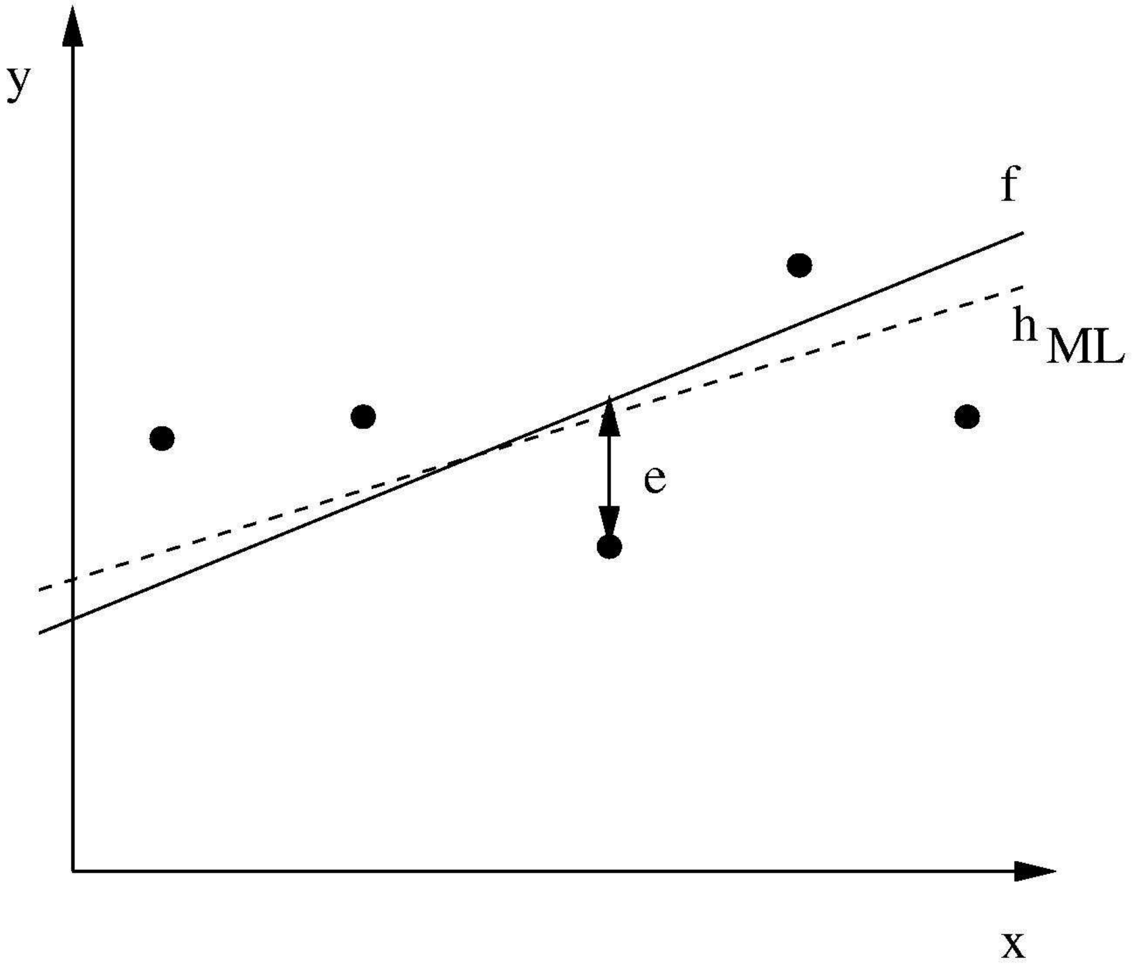
Then

$$P(h|D) = \begin{cases} \frac{1}{|VS_{H,D}|} & \text{if } h \text{ is consistent with } D \\ 0 & \text{otherwise} \end{cases}$$

Evolution of Posterior Probabilities



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=0

Then the maximum likelihood hypothesis h_{ML} is the one that minimizes the sum of squared errors:

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

Maximum likelihood hypothesis:

$$
\begin{aligned}
h_{ML} &= \underset{h \in H}{\operatorname{argmax}}\, p(D|h) = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} p(d_i|h) \\
&= \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{d_i - h(x_i)}{\sigma}\right)^2}
\end{aligned}
$$

Maximize natural log of this instead . . .

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

Most Probable Classification of New Instances

So far we've sought the most probable *hypothesis* given the data D (i.e., h_{MAP})

Given new instance x, what is its most probable *classification*? Not $h_{MAP}(x)$!

Consider:

- Three possible hypotheses:

$$P(h_1|D) = .4,\ P(h_2|D) = .3,\ P(h_3|D) = .3$$

- Given new instance x,

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

- What's most probable classification of x?

Bayes Optimal Classifier

Bayes optimal classification:

$$\arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D)$$

Example:

$$P(h_1|D) = .4, \quad P(-|h_1) = 0, \quad P(+|h_1) = 1$$
$$P(h_2|D) = .3, \quad P(-|h_2) = 1, \quad P(+|h_2) = 0$$
$$P(h_3|D) = .3, \quad P(-|h_3) = 1, \quad P(+|h_3) = 0$$

therefore

$$\sum_{h_i \in H} P(+|h_i)P(h_i|D) = .4$$

$$\sum_{h_i \in H} P(-|h_i)P(h_i|D) = .6$$

and

$$\arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D) = -$$

Gibbs Classifier

Bayes optimal classifier is hopelessly inefficient

Gibbs algorithm:

1. Choose one hypothesis at random, 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

$$E[error_{Gibbs}] \leq 2 \times E[error_{BayesOptimal}]$$

Naive Bayes Classifier

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

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

$$v_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j | a_1, a_2 \ldots a_n)$$

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

Naive Bayes assumption:

$$P(a_1, a_2 \ldots a_n | v_j) = \prod_i P(a_i | v_j)$$

which gives

Naive Bayes classifier:

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

Naive Bayes Algorithm

Naive_Bayes_Learn(*examples*)

For each target value v_j

- $\hat{P}(v_j) \leftarrow$ estimate $P(v_j)$
- For each attribute value a_i of each attribute a
  $\hat{P}(a_i|v_j) \leftarrow$ estimate $P(a_i|v_j)$

Classify_New_Instance(x)

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

Naive Bayes: Example

Consider *PlayTennis* again, and new instance

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

Want to compute:

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

$$P(y)\ P(sun|y)\ P(cool|y)\ P(high|y)\ P(strong|y) = .005$$

$$P(n)\ P(sun|n)\ P(cool|n)\ P(high|n)\ P(strong|n) = .021$$

$$\rightarrow v_{NB} = n$$

Naive Bayes: Subtleties (1)

Conditional independence assumption is often violated

$$P(a_1, a_2 \ldots a_n | v_j) = \prod_i P(a_i | v_j)$$

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

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

Naive Bayes posteriors often unrealistically close to 1 or 0

Naive Bayes: Subtleties (2)

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

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

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

Typical solution is m-estimate for $\hat{P}(a_i|v_j)$

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

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)

Learning to Classify Text

Why?

- Learn which news articles are of interest
- Learn to classify web pages by topic

Naive Bayes is among most effective algorithms

What attributes shall we use to represent text documents?

Learning to Classify Text

Target concept $Interesting? : Document \rightarrow \{+, -\}$

1. Represent each document by vector of words: one attribute per word position in document
2. Learning: Use training examples to estimate
   - $P(+)$
   - $P(-)$
   - $P(doc|+)$
   - $P(doc|-)$

Naive Bayes conditional independence assumption

$$P(doc|v_j) = \prod_{i=1}^{length(doc)} P(a_i = w_k|v_j)$$

where $P(a_i = w_k|v_j)$ is probability that word in position i is w_k, given v_j

One more assumption:
$P(a_i = w_k|v_j) = P(a_m = w_k|v_j), \forall i, m$

LEARN_NAIVE_BAYES_TEXT($Examples, V$)

1. Collect all words & tokens that occur in Examples

- $Vocabulary \leftarrow$ all distinct words & tokens in $Examples$

2. Compute all probabilities $P(v_j)$ and $P(w_k|v_j)$

- For each target value v_j in V do
  - $docs_j \leftarrow Examples$ for which the target value is v_j
  - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
  - $Text_j \leftarrow$ concatenate all members of $docs_j$
  - $n \leftarrow$ total number of words in $Text_j$ (counting duplicate words multiple times)
  - for each word w_k in $Vocabulary$
    * $n_k \leftarrow$ number of times word w_k occurs in $Text_j$
    * $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

CLASSIFY_NAIVE_BAYES_TEXT(*Doc*)

- *positions* $\leftarrow$ all word positions in *Doc* that contain tokens found in *Vocabulary*
- Return v_{NB}, where

$$v_{NB} = \operatorname*{argmax}_{v_j \in V} P(v_j) \prod_{i \in positions} P(a_i | v_j)$$

Example: 20 Newsgroups

Given 1000 training documents from each group

Learn to classify new documents according to which newsgroup it came from

comp.graphics	misc.forsale
comp.os.ms-windows.misc	rec.autos
comp.sys.ibm.pc.hardware	rec.motorcycles
comp.sys.mac.hardware	rec.sport.baseball
comp.windows.x	rec.sport.hockey
alt.atheism	sci.space
soc.religion.christian	sci.crypt
talk.religion.misc	sci.electronics
talk.politics.mideast	sci.med
talk.politics.misc	talk.politics.guns

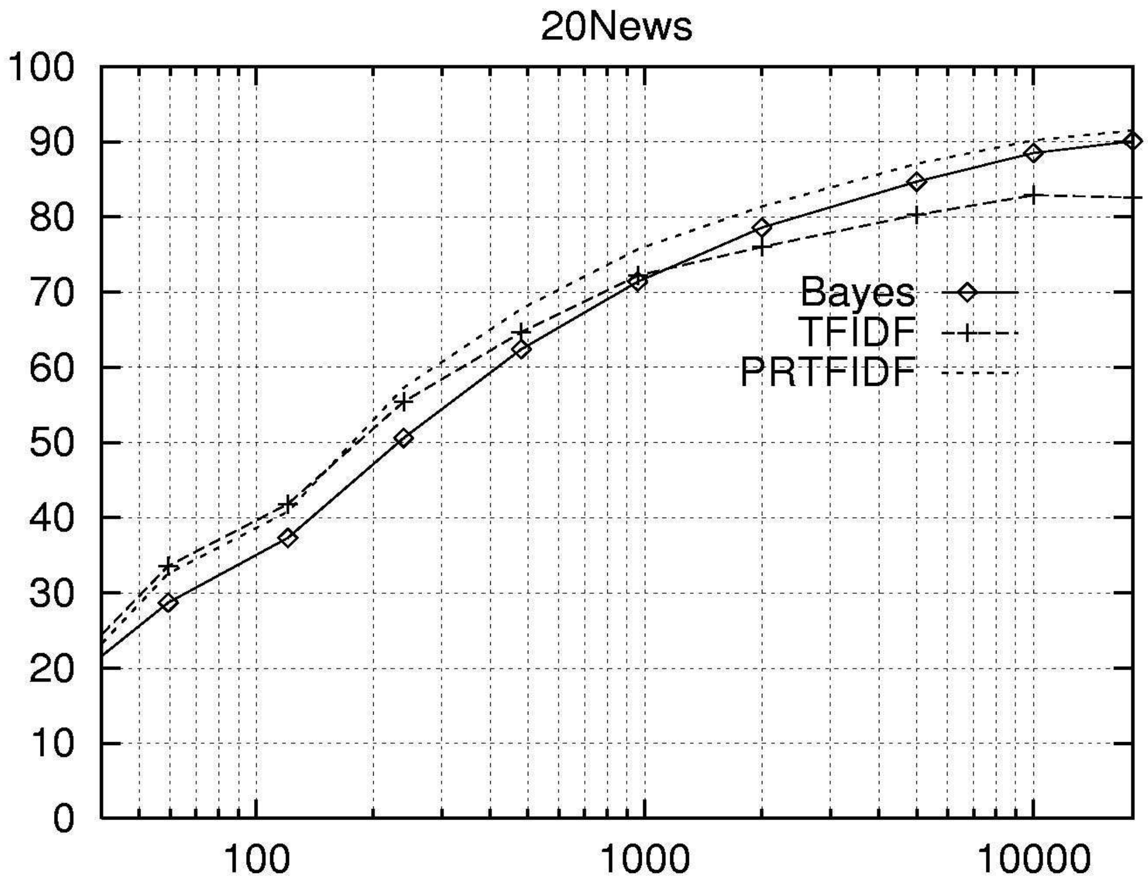
Naive Bayes: 89% classification accuracy

Article from rec.sport.hockey

Path: cantaloupe.srv.cs.cmu.edu!...
From: xxx@yyy.zzz.edu (John Doe)
Subject: Re: This year's biggest and worst (opinion)
Date: 5 Apr 93 09:53:39 GMT

I can only comment on the Kings, but the most obvious candidate for pleasant surprise is Alex Zhitnik. He came highly touted as a defensive defenseman, but he's clearly much more than that. Great skater and hard shot (though wish he were more accurate). In fact, he pretty much allowed the Kings to trade away that huge defensive liability Paul Coffey. Kelly Hrudey is only the biggest disappointment if you thought he was any good to begin with. But, at best, he's only a mediocre goaltender. A better choice would be Tomas Sandstrom, though not through any fault of his own, but because some thugs in Toronto decided

Learning Curve for 20 Newsgroups



Accuracy vs. Training set size (1/3 withheld for test)

Bayesian Networks

Interesting because:

- Naive Bayes assumption of conditional independence too restrictive
- But it's intractable without some such assumptions ...
- Bayesian networks describe conditional independence among *subsets* of variables
- This allows combining 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 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

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

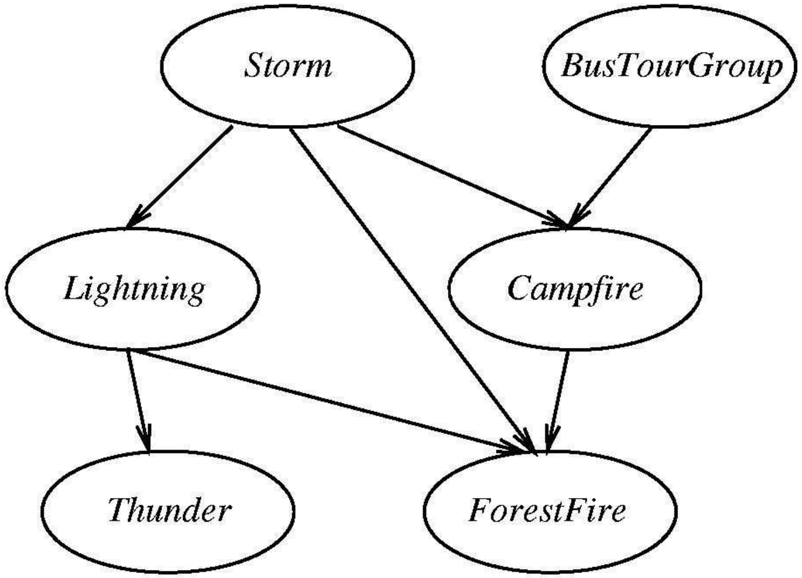
Example: $Thunder$ is conditionally independent of $Rain$, given $Lightning$

$$P(Thunder|Rain, Lightning) = P(Thunder|Lightning)$$

Naive Bayes uses cond. indep. to justify

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

Bayesian Network



	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

Network represents a set of conditional independence assertions:

Each node is conditionally independent of its nondescendants, given its parents

Network represents joint probability distribution over all variables

- E.g., $P(Storm, BusTourGroup, \ldots, ForestFire)$
- In general,

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

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

- So joint distribution is fully defined by graph, plus the $P(y_i | Parents(Y_i))$
- What is the graph of Naive Bayes?

Inference in Bayesian Networks

How can one infer the (probabilities of) values of one or more network variables, given observed values of others?

- Bayes net contains all information needed for this inference
- 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 Bayesian Networks

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 no missing values,
it's as easy as training a Naive Bayes classifier

The EM Algorithm

Suppose structure known, variables partially observable

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

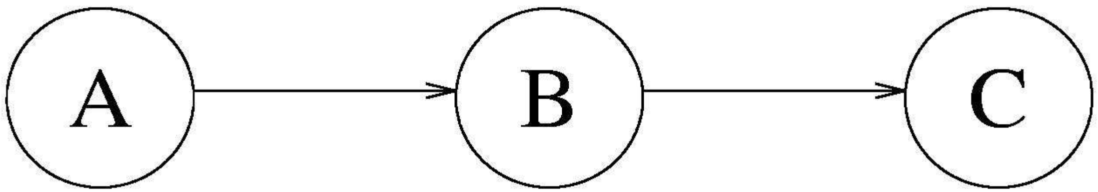
Initialize parameters ignoring missing information

Repeat until convergence:

E step: Calculate expected vals of unobserved variables, assuming current parameter values

M step: Calculate new parameter values to maximize probability of data (observed & estimated)

Example



Examples:

A	B	C
0	1	1
1	0	0
1	1	1
1	?	0

Initialization: $P(B|A) =$ $P(C|B) =$

$P(A) =$ $P(B|\neg A) =$ $P(C|\neg B) =$

E-step: $P(? = 1) = P(B|A, \neg C) = \frac{P(A,B,\neg C)}{P(A,\neg C)} = \ldots = 0$

M-step: $P(B|A) =$ $P(C|B) =$

$P(A) =$ $P(B|\neg A) =$ $P(C|\neg B) =$

E-step: $P(? = 1) = 0$ **(converged)**

Unknown Structure

Search:

- Initial state: empty network, prior network
- Operators: Add arc, delete arc, reverse arc
- Evaluation: Posterior probability

Bayesian Learning: Summary

- Optimal prediction
- Naive Bayes learner
- Text classification
- Bayesian networks
- EM algorithm