Bayesian Learning

- Bayes Theorem
- MAP, ML hypotheses
- MAP learners
- Minimum description length principle
- Bayes optimal classifier
- Naive Bayes learner
- Example: Learning over text data
- Bayesian belief networks
- Expectation Maximization algorithm

Two Roles for Bayesian Methods

Provides practical learning algorithms:

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

Provides useful conceptual framework

- Provides "gold standard" for evaluating other learning algorithms
- Additional insight into Occam's razor

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

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

Generally want the most probable hypothesis given the training data

Maximum a posteriori hypothesis h_{MAP}:

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

If assume $P(h_i) = P(h_j)$ then can further simplify, and choose the *Maximum likelihood* (ML) hypothesis

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

Bayes Theorem

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.

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

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:

  $$\begin{aligned} P(A \vee B) &= P(A) + P(B) - P(A \wedge B) \\ &= P(A) + P(B) \end{aligned}$$

  if A and B are mut. ex.

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

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

Relation to Concept Learning

Consider our usual concept learning task

- instance space X, hypothesis space H, training examples D
- consider the FindS learning algorithm (outputs most specific hypothesis from the version space $VS_{H,D}$)

What would Bayes rule produce as the MAP hypothesis?

Does $FindS$ output a MAP hypothesis??

Relation to Concept Learning

Assume fixed set of instances $\langle x_1, \ldots, x_m \rangle$
Assume D is the set of classifications
$D = \langle c(x_1), \ldots, c(x_m) \rangle$
Choose $P(D|h)$

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

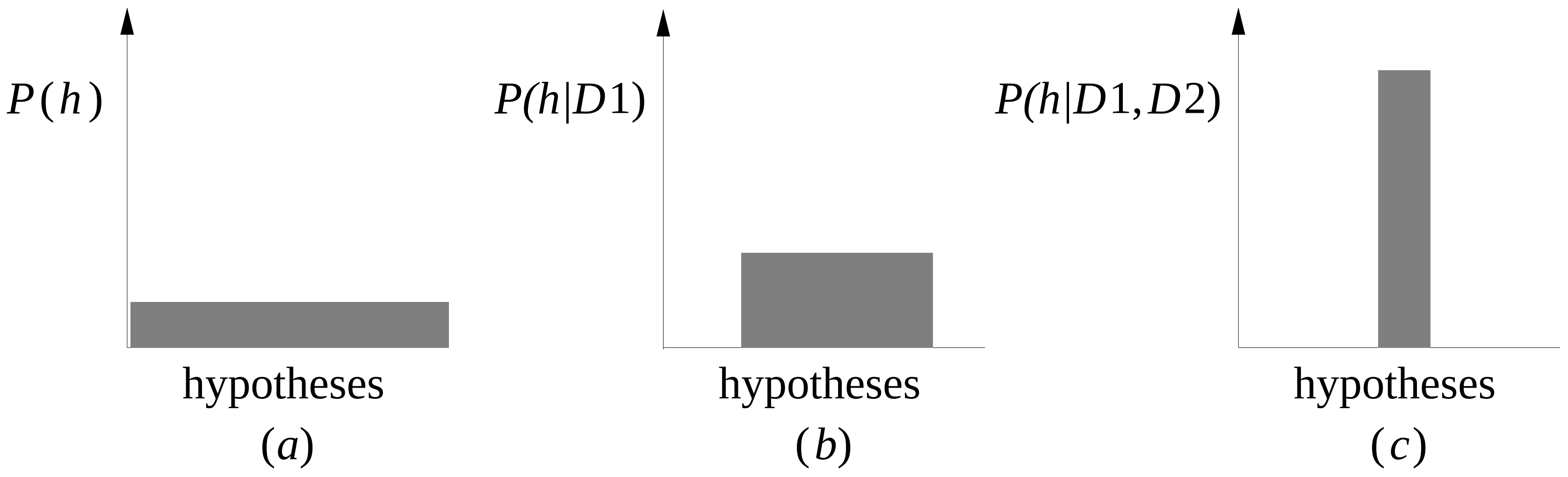
Choose $P(h)$ to be *uniform* distribution

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

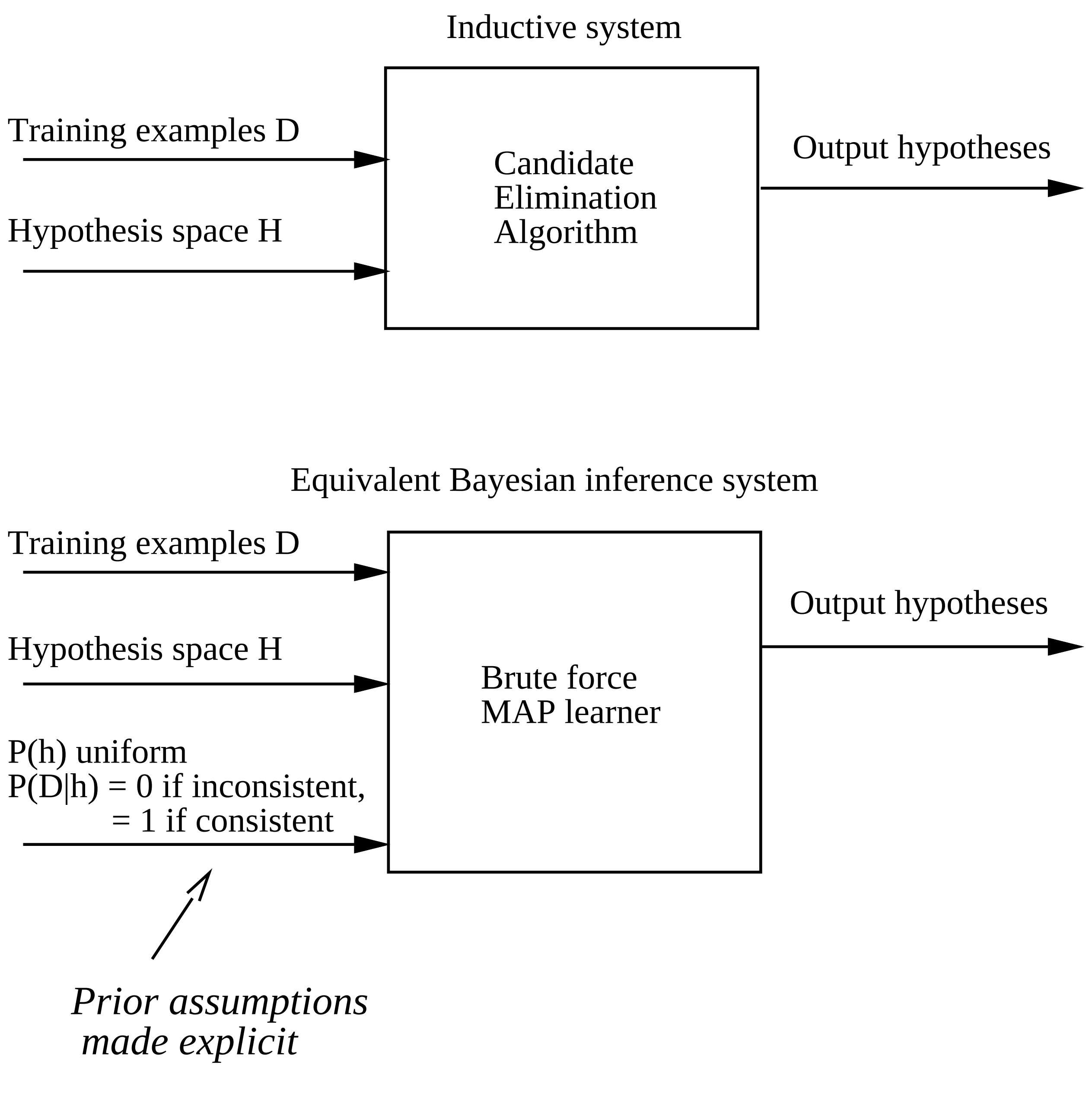
Then,

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

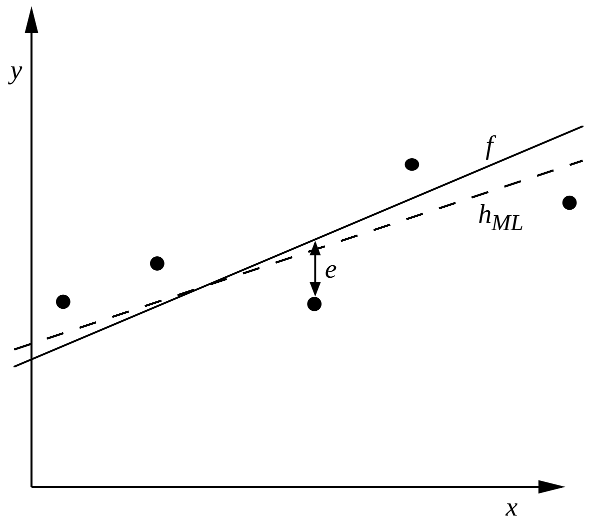
Evolution of Posterior Probabilities



Characterizing Learning Algorithms by Equivalent MAP Learners



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:

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

Learning A Real Valued Function

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

Maximize natural log of this instead...

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

Minimum Description Length Principle

Occam's razor: prefer the shortest hypothesis

MDL: prefer the hypothesis h that minimizes

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

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

Fact from information theory:

The optimal (shortest expected coding length) code for an event with probability p is $-\log_2 p$ bits.

$$\begin{aligned} h_{MAP} &= \arg\max_{h \in H} P(D|h)P(h) \\ &= \arg\max_{h \in H} \log_2 P(D|h) + \log_2 P(h) \\ &= \arg\min_{h \in H} -\log_2 P(D|h) - \log_2 P(h) \end{aligned}$$

Interpret as follows:

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

$\rightarrow$ prefer the hypothesis that minimizes

$$length(h) + length(misclassifications)$$

$\rightarrow h_{MAP}$ is Occam's h
$\rightarrow h_{ML}$ is Occam's h when the prior is uninformative

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*?
$\rightarrow$ If predictions are not mut. ex. then $h_{MAP}(x)$ is not the most probable classification!

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

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

Example:

$$\begin{aligned}
P(h_1|D) = .4,\ P(-|h_1) = 0,\ P(+|h_1) = 1 \\
P(h_2|D) = .3,\ P(-|h_2) = 1,\ P(+|h_2) = 0 \\
P(h_3|D) = .3,\ P(-|h_3) = 1,\ P(+|h_3) = 0
\end{aligned}$$

therefore

$$\begin{aligned}
\sum_{h_i \in H} P(+|h_i) P(h_i|D) &= .4 \\
\sum_{h_i \in H} P(-|h_i) P(h_i|D) &= .6
\end{aligned}$$

and

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

Problem: Can be too expensive if $|H|$ large or infinite.

Naive Bayes Classifier

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
- When you need a quick and dirty solution from scratch.

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: $v_{NB} =$

$$= \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i | v_j)$$

Naive Bayes Algorithm

```
Naive_Bayes_Learn($examples$)
    For each possible 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:

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

$$P(y)\ P(sun|y)\ P(cool|y)\ P(high|y)\ \times$$
$$P(strong|y) = .005$$
$$P(n)\ P(sun|n)\ P(cool|n)\ P(high|n)\ \times$$
$$P(strong|n) = .021$$

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

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

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

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

- see [Domingos & Pazzani, 1996] for analysis

Naive Bayes: Subtleties

2. what if none of the training instances with target value v_j have attribute value a_i? Then

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

Typical solution is Bayesian 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

E.g.:

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

Naive Bayes is among most effective (1st generation) learning 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

Also assume position independence:
$P(a_i = w_k|v_j) = P(a_m = w_k|v_j), \forall i, m$

Twenty 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!das-news.harvard.edu!ogicse!uwm.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
```

Learn_naive_Bayes_text($Examples, V$)

```
For each target value $v_j$ in $V$ do
  • $docs_j \leftarrow$ subset of $Examples$ for which the target value is $v_j$
  • $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
  • $Text_j \leftarrow$ a single document created by concatenating 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

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

Bayesian Belief Networks

Also called Bayes Nets.
Interesting because:

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

$\rightarrow$ 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

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

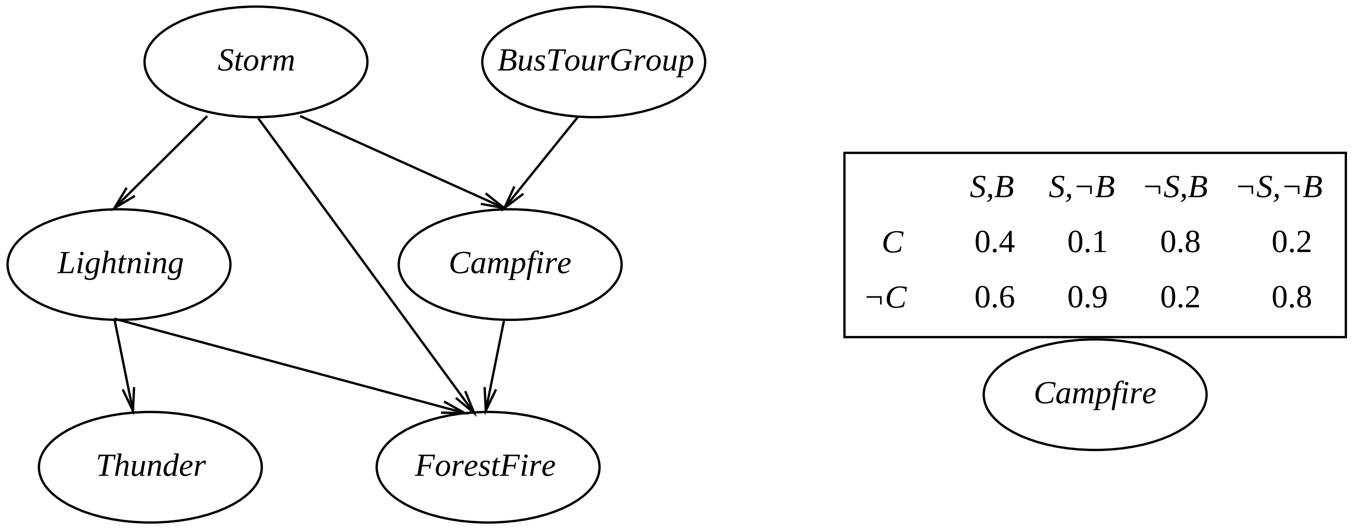
Example: $Thunder$ is conditionally independent of $Rain$, given $Lightning$

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

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 Belief Network

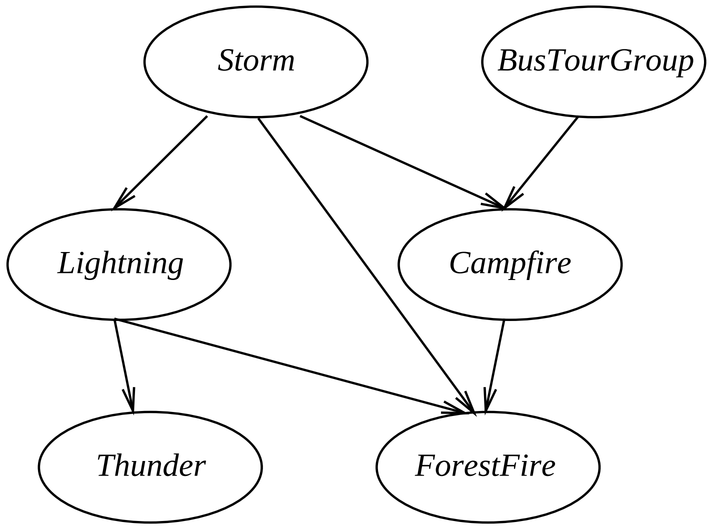


	S,B	*S,¬B*	*¬S,B*	*¬S,¬B*
C	0.4	0.1	0.8	0.2
¬C	0.6	0.9	0.2	0.8

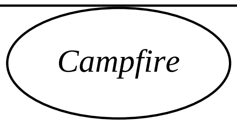
Network represents a set of conditional independence assertions:

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

Bayesian Belief Network



	S,B	*S,¬B*	*¬S,B*	*¬S,¬B*
C	0.4	0.1	0.8	0.2
¬C	0.6	0.9	0.2	0.8



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 $P(y_i|Parents(Y_i))$

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 the information needed to make this inference using repeated application of the axioms of probability theory.
- If only one variable with unknown value, easy to infer it
- In general case, problem is NP hard

In practice, can succeed in many cases

- Exact inference methods work well for some (sparse) network structures
- Monte Carlo methods "simulate" the network randomly to calculate approximate solutions

Learning of 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 observe all variables

- Then it's easy as training a Naive Bayes classifier

Learning Bayes Nets

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 $P(D|h)$

Gradient Ascent for Bayes Nets

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

$$w_{ijk} = P(Y_i = y_{ij} | Parents(Y_i) =$$

$$= \text{the list } u_{ik} \text{ of values})$$

e.g., if $Y_i = Campfire$, then u_{ik} might be $\langle Storm = T, BusTourGroup = F \rangle$

Perform gradient ascent by repeatedly

1. update all w_{ijk} using training data D

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

2. then, renormalize the w_{ijk} to assure

- $\sum_j w_{ijk} = 1$
- $0 \leq w_{ijk} \leq 1$

More on Learning Bayes Nets

EM algorithm can also be used. Repeatedly:

1. Calculate probabilities of unobserved variables, assuming h
2. Calculate new w_{ijk} to maximize $E[\ln P(D|h)]$ where D now includes both observed and (calculated probabilities of) unobserved variables

When structure unknown...

- Can use greedy search to add/substract edges and nodes
- Active research topic

Summary: Bayesian Belief Networks

- 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
  - Extend to first-order instead of propositional systems
  - More effective inference methods
  - ...

Expectation Maximization (EM)

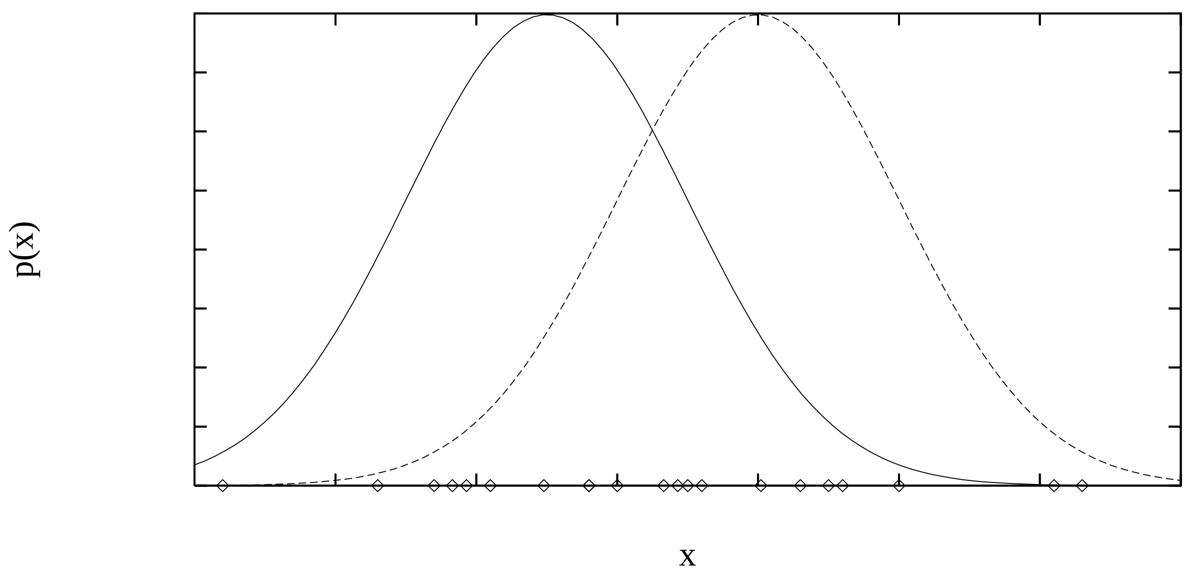
When to use:

- Data is only partially observable
- Unsupervised clustering (target value unobservable)
- Supervised learning (some instance attributes unobservable)

Some uses:

- Train Bayesian Belief Networks
- Unsupervised clustering (AUTOCLASS)
- Learning Hidden Markov Models
- Learning parameters of any stochastic process.

Generating Data from Mixture of k Gaussians



Each instance x generated by

1. Choosing one of the k Gaussians with uniform probability
2. Generating an instance at random according to that Gaussian

EM for Estimating k Means

Given:

- Instances from X generated by mixture of k Gaussian distributions
- Unknown means $\langle \mu_1, \ldots, \mu_k \rangle$ of the k Gaussians
- Don't know which instance x_i was generated by which Gaussian

Determine:

- Maximum likelihood estimates of $\langle \mu_1, \ldots, \mu_k \rangle$

Think of full description of each instance as $y_i = \langle x_i, z_{i1}, z_{i2} \rangle$, where

- z_{ij} is 1 if x_i generated by jth Gaussian
- x_i observable
- z_{ij} unobservable

EM for Estimating k Means

EM Algorithm: Pick random initial $h = \langle \mu_1, \mu_2 \rangle$, then iterate

E step: Calculate the expected value $E[z_{ij}]$ of each hidden variable z_{ij}, assuming the current hypothesis $h = \langle \mu_1, \mu_2 \rangle$ holds.

$$E[z_{ij}] = \frac{p(x = x_i | \mu = \mu_j)}{\Sigma_{n=1}^{2} p(x = x_i | \mu = \mu_n)} = \frac{e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)^2}}{\Sigma_{n=1}^{2} e^{-\frac{1}{2\sigma^2}(x_i - \mu_n)^2}}$$

M step: Calculate a new maximum likelihood hypothesis $h' = \langle \mu'_1, \mu'_2 \rangle$, assuming the value taken on by each hidden variable z_{ij} is its expected value $E[z_{ij}]$ calculated above. Replace $h = \langle \mu_1, \mu_2 \rangle$ by $h' = \langle \mu'_1, \mu'_2 \rangle$.

$$\mu_j \leftarrow \frac{\Sigma_{i=1}^{m} E[z_{ij}] \; x_i}{\Sigma_{i=1}^{m} E[z_{ij}]}$$

EM Algorithm

Converges to local maximum likelihood h
and provides estimates of hidden variables z_{ij}

In fact, local maximum in $E[\ln P(Y|h)]$

- Y is complete (observable plus unobservable variables) data
- Expected value is taken over possible values of unobserved variables in Y

General EM Method

Define likelihood function $Q(h'|h)$ which calculates $Y = X \cup Z$ using observed X and current parameters h to estimate Z

$$Q(h'|h) \leftarrow E[\ln P(Y|h')|h, X]$$

EM Algorithm:

Estimation (E) step: Calculate $Q(h'|h)$ using the current hypothesis h and the observed data X to estimate the probability distribution over Y.

$$Q(h'|h) \leftarrow E[\ln P(Y|h')|h, X]$$

Maximization (M) step: Replace hypothesis h by the hypothesis h' that maximizes this Q function.

$$h \leftarrow \underset{h'}{\operatorname{argmax}}\, Q(h'|h)$$