

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

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.

Welcome to conditional probabilities!

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

Thomas Bayes



Thomas Bayes (c. 1701 – 7 April 1761) was an English mathematician and Presbyterian minister, known for having formulated a specific case of the theorem that bears his name: Bayes' theorem. Bayes never published what would eventually become his most famous accomplishment; his notes were edited and published after his death by Richard Price.

Bayes Theorem

The most probable hypothesis given the data D .

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

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

$$P(\text{cancer}) =$$

$$P(\neg \text{cancer}) =$$

$$P(+|\text{cancer}) =$$

$$P(-|\text{cancer}) =$$

$$P(+|\neg \text{cancer}) =$$

$$P(-|\neg \text{cancer}) =$$

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

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

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 $\langle x_1, \dots, x_m \rangle$

Assume D is the set of classifications $D = \langle c(x_1), \dots, c(x_m) \rangle$

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

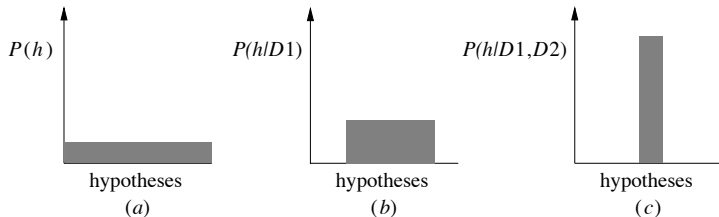
Because $VS_{H,D}$ are H consistent with D ,

- ▶ $P(D) = \frac{|VS_{H,D}|}{|H|}$

Then,

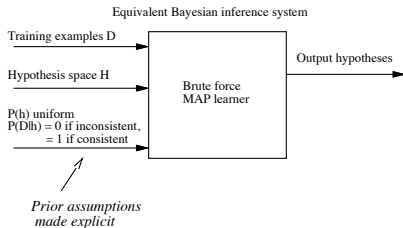
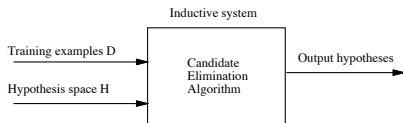
$$P(h|D) = \begin{cases} \frac{1}{|VS_{H,D}|} & \text{if } h \text{ is consistent with } D \\ 0 & \text{otherwise} \end{cases}$$

Evolution of Posterior Probabilities

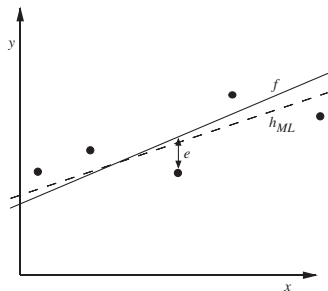


Posterior prob. increases as there are less hypothesis in version space.

Characterizing Learning Algorithms by Equivalent MAP Learners



Learning A Real Valued Function

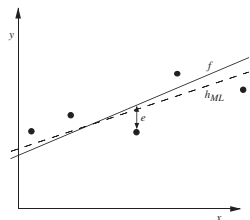


Consider any real-valued target function f

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

Learning A Real Valued Function



$$d_i = f(x_i) + e_i$$

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

$$\begin{aligned}h_{ML} &= \operatorname{argmax}_{h \in H} p(D|h) \\ &= \operatorname{argmax}_{h \in H} \prod_{i=1}^m p(d_i|h) \\ \text{Now, } d_i &\text{ is normal } (\mu = f(x) = h(x), \sigma^2) \\ &= \operatorname{argmax}_{h \in H} \prod_{i=1}^m \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{d_i - h(x_i)}{\sigma}\right)^2}\end{aligned}\tag{1}$$

Maximize natural log of this instead...

Learning A Real Valued Function

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

Learning A Real Valued Function

- ▶ Based on normality of the error (convenient and real)
- ▶ We've only considered noise in target value
- ▶ Gets more and more complex if we add noise in data measurements

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

Want to train neural network to output a *probability* given x_i (not a 0 or 1)

In this case can show

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

Weight update rule for a sigmoid unit:

$$w_{jk} \leftarrow w_{jk} + \Delta w_{jk}$$

where

$$\Delta w_{jk} = \eta \sum_{i=1}^m (d_i - h(x_i)) x_{ijk}$$

Minimum Description Length Principle

Occam's razor: prefer the shortest hypothesis

MDL: prefer the hypothesis h that minimizes

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

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

$$\begin{aligned}h_{MAP} &= \arg \max_{h \in H} P(D|h)P(h) \\ &= \arg \max_{h \in H} \log_2 P(D|h) + \log_2 P(h) \\ &= \arg \min_{h \in H} -\log_2 P(D|h) - \log_2 P(h)\end{aligned}\quad (2)$$

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

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

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*?

- ▶ $h_{MAP}(x)$ 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

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 provides best result, but can be expensive if many hypotheses.

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:

$$E[\text{error}_{Gibbs}] \leq 2E[\text{error}_{BayesOptimal}]$$

Suppose correct, uniform prior distribution over H , then

- ▶ Pick any hypothesis from VS, with uniform probability
- ▶ Its expected error no worse than twice Bayes optimal

Naive Bayes Classifier

Along with decision trees, neural networks, nearest nbr, 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

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:

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

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

Naïve Bayes Classifier

Naive Bayes assumption:

conditional independence : if (A ind B) (3)

$$\Rightarrow P(A|B, C) = P(A|C)$$

$$\Rightarrow P(A, B|C) = P(A|C)P(B|C)$$

$$P(a_1, a_2 \dots a_n | v_j) = \prod_i P(a_i | v_j) \quad (4)$$

which gives

Naïve 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

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

Want to compute:

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

$$V = \{y, n\}$$

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

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

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

Naive Bayes: Subtleties

1. Conditional independence assumption is often violated

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

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

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

- ▶ see [Domingos & Pazzani, 1996] for analysis
- ▶ 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

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

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

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 $\hat{P}(a_i|v_j)$ (e.g. $\frac{1}{k}$)
- ▶ 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: $P(a_i = w_k|v_j) = P(a_m = w_k|v_j), \forall i, m$

Learning to Classify Text

Learn_naive_Bayes_text(*Examples*, V)

1. collect all words and other tokens that occur in *Examples*
 - ▶ *Vocabulary* \leftarrow all distinct words and other tokens in *Examples*
2. calculate the required $P(v_j)$ and $P(w_k|v_j)$ probability terms
 - ▶ 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 $Text_j$
 - ▶ $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

Learning to Classify Text

Classify_naive_Bayes_text(*Doc*)

- ▶ *positions* ← all word positions in *Doc* that contain tokens found in *Vocabulary*
- ▶ Return v_{NB} , where

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

Twenty NewsGroups

Given 1000 training documents from each group
Learn to classify new documents according to which
newsgroup it came from

comp.graphics
comp.os.ms-windows.misc
comp.sys.ibm.pc.hardware
comp.sys.mac.hardware
comp.windows.x

misc.forsale
rec.autos
rec.motorcycles
rec.sport.baseball
rec.sport.hockey

alt.atheism
soc.religion.christian
talk.religion.misc
talk.politics.mideast
talk.politics.misc
talk.politics.guns

sci.space
sci.crypt
sci.electronics
sci.med

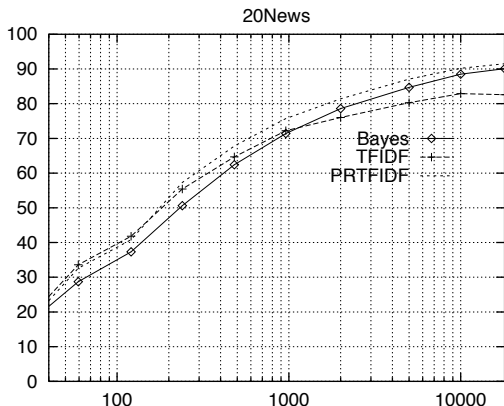
Naive Bayes: 89% classification accuracy

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

Learning Curve for 20 Newsgroups



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

Bayesian Belief Networks

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
- allows combining prior knowledge about (in)dependencies among variables with observed training data

(also called Bayes Nets)

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

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

Conditional Independence

Example

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

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

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 Networks

Example

I'm at work, neighbor John calls to say my alarm is ringing, but neighbor Mary doesn't call. Sometimes it's set off by minor earthquakes. Is there a burglar?

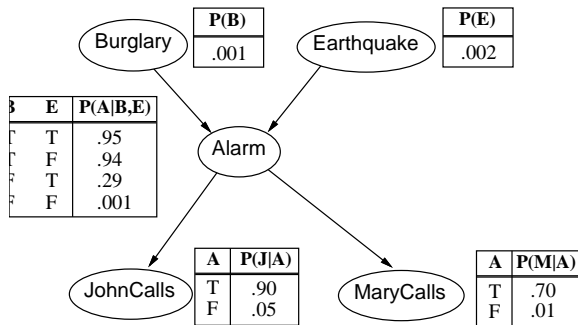
Variables: *Burglar*, *Earthquake*, *Alarm*, *JohnCalls*, *MaryCalls*

Network topology reflects “causal” knowledge:

- ▶ A burglar can set the alarm off
- ▶ An earthquake can set the alarm off
- ▶ The alarm can cause Mary to call
- ▶ The alarm can cause John to call

Bayesian Belief Networks

Example



Compactness

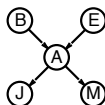
A CPT for Boolean X_i with k Boolean parents has 2^k rows for the combinations of parent values

Each row requires one number p for $X_i = \text{true}$ (the number for $X_i = \text{false}$ is just $1 - p$)

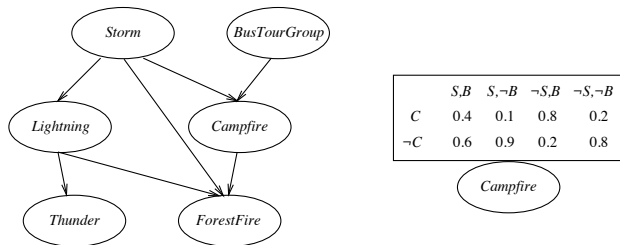
If each variable has no more than k parents, the complete network requires $O(n \cdot 2^k)$ numbers

I.e., grows linearly with n , vs. $O(2^n)$ for the full joint distribution

For burglary net, $1 + 1 + 4 + 2 + 2 = 10$ numbers (vs. $2^5 - 1 = 31$)



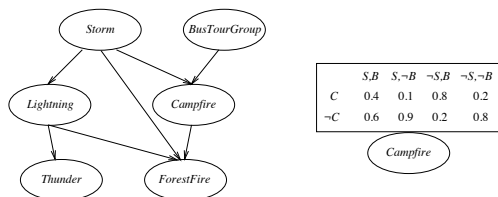
Bayesian Belief Network



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



Represents joint probability distribution over all variables

- ▶ e.g., $P(\text{Storm}, \text{BusTourGroup}, \dots, \text{ForestFire})$
- ▶ in general,

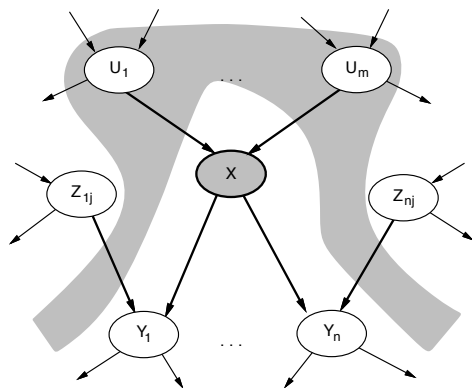
$$P(y_1, \dots, y_n) = \prod_{i=1}^n P(y_i | \text{Parents}(Y_i))$$

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

- ▶ so, joint distribution is fully defined by graph, plus the $P(y_i | \text{Parents}(Y_i))$

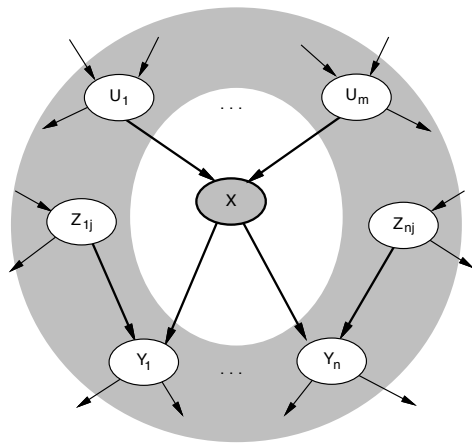
Semantics

Local semantics: each node is conditionally independent of its nondescendants given its parents



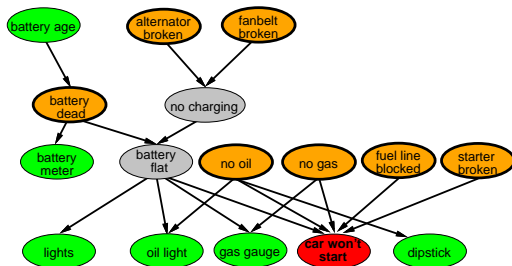
Local Semantics

Each node is conditionally independent of all others given its **Markov blanket**: parents + children + children's parents



example

- ▶ Initial evidence: car won't start
- ▶ Testable variables (green), "broken, so fix it" variables (orange)
- ▶ Hidden variables (gray) ensure sparse structure, reduce parameters



Compact Conditional Distributions

- ▶ CPT grows exponentially with number of parents
- ▶ CPT becomes infinite with continuous-valued parent or child

Solution: **canonical** distributions that are defined compactly
Deterministic nodes are the simplest case:

$X = f(\text{Parents}(X))$ for some function f

E.g., Boolean functions

NorthAmerican = Canadian OR US OR Mexican

E.g., numerical relationships among continuous variables

$$\frac{\partial \text{Level}}{\partial t} = \text{inflow} + \text{precipitation} - \text{outflow} - \text{evaporation}$$

Compact Conditional Distributions

Noisy OR: Modeling Multiple Non-Interacting Causes

- ▶ Parents $U_1 \dots U_k$ include all causes (can add leak node)
- ▶ Independent failure probability q_i for each cause alone
 $\implies P(X|U_1 \dots U_j, \neg U_{j+1} \dots \neg U_k) = 1 - \prod_{i=1}^j q_i$

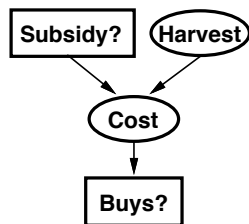
<i>Cold</i>	<i>Flu</i>	<i>Malaria</i>	$P(\text{Fever})$	$P(\neg \text{Fever})$
F	F	F	0.0	1.0
F	F	T	0.9	0.1
F	T	F	0.8	0.2
F	T	T	0.98	$0.02 = 0.2 * 0.1$
T	F	F	0.4	0.6
T	F	T	0.94	$0.06 = 0.6 * 0.1$
T	T	F	0.88	$0.12 = 0.6 * 0.2$
T	T	T	0.988	$0.012 = 0.6 * 0.2 * 0.1$

Number of parameters *linear* in number of parents

Hybrid (discrete+continuous) networks

Discrete (*Subsidy?* and *Buys?*); continuous (*Harvest* and *Cost*)

- ▶ Option 1:
discretization—possibly large errors, large CPTs
- ▶ Option 2: finitely parameterized canonical families
- ▶ Continuous variable, discrete+continuous parents (e.g., *Cost*)
- ▶ Discrete variable, continuous parents (e.g., *Buys?*)



Continuous child variables

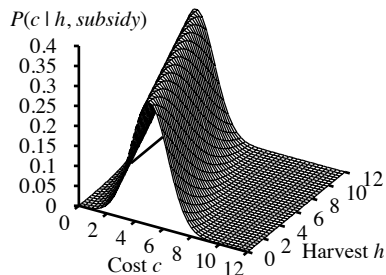
Need one **conditional density** function for child variable given continuous parents, for each possible assignment to discrete parents

Most common is the **linear Gaussian** model, e.g.,:

$$\begin{aligned} P(\text{Cost} = c | \text{Harvest} = h, \text{Subsidy?} = \text{true}) \\ &= N(a_t h + b_t, \sigma_t)(c) \\ &= \frac{1}{\sigma_t \sqrt{2\pi}} \exp\left(-\frac{1}{2} \left(\frac{c - (a_t h + b_t)}{\sigma_t}\right)^2\right) \end{aligned}$$

Mean *Cost* varies linearly with *Harvest*, variance is fixed
Linear variation is unreasonable over the full range, but works OK if the *likely* range of *Harvest* is narrow

Continuous child variables



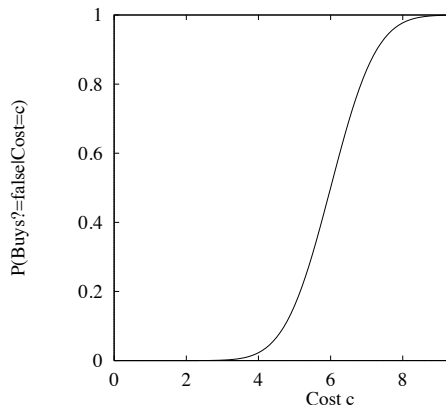
All-continuous network with LG distributions \implies full joint distribution is a multivariate Gaussian

Discrete+continuous LG network is a **conditional Gaussian** network

i.e., a multivariate Gaussian over all continuous variables for each combination of discrete variable values

Discrete variable w/ continuous parents

Probability of *Buys?* given *Cost* should be a “soft” threshold:



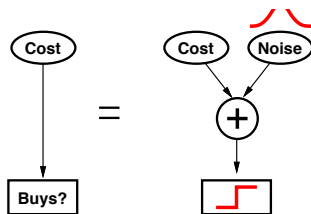
Probit distribution uses integral of Gaussian:

$$\Phi(x) = \int_{-\infty}^x N(0, 1)(x) dx$$

$$P(\text{Buys?} = \text{true} | \text{Cost} = c) = \Phi((-c + \mu)/\sigma)$$

Why the probit?

1. It's sort of the right shape
2. Can view as hard threshold whose location is subject to noise

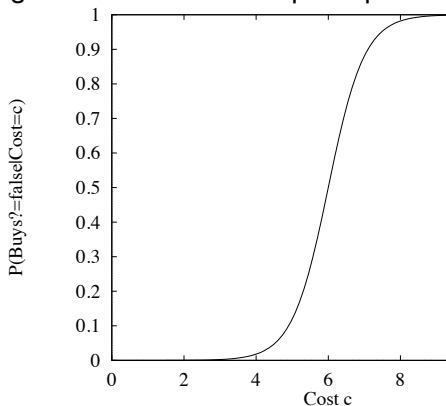


Discrete variable contd.

Sigmoid (or **logit**) distribution also used in neural networks:

$$P(\text{Buys?} = \text{true} | \text{Cost} = c) = \frac{1}{1 + \exp(-2\frac{-c+\mu}{\sigma})}$$

Sigmoid has similar shape to probit but much longer tails:



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
- ▶ 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 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.

$i = \text{node}, j = \text{node value}, k = \text{parents configuration}$

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

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

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...

- ▶ Algorithms use greedy search to add/subtract 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)

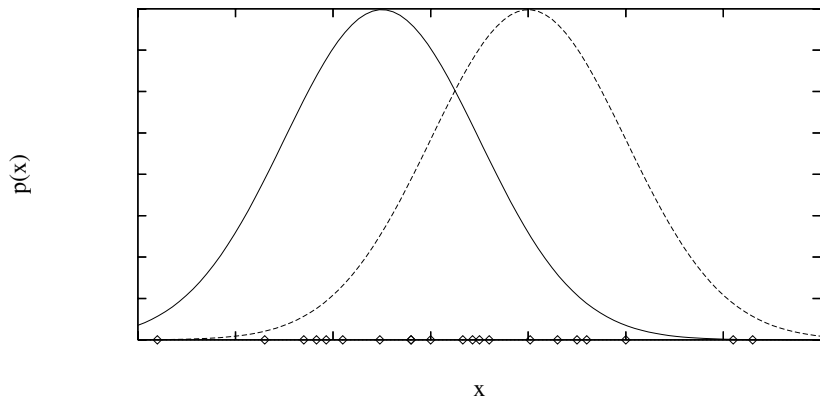
When to use:

- ▶ 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

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 $\langle \mu_1, \dots, \mu_k \rangle$ of the k Gaussians
- ▶ Don't know which instance x_i was generated by which Gaussian

Determine:

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

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 j th 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.

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

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 $h' = \langle \mu'_1, \mu'_2 \rangle$.

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

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 Problem

Given:

- ▶ Observed data $X = \{x_1, \dots, x_m\}$
- ▶ Unobserved data $Z = \{z_1, \dots, z_m\}$
- ▶ Parameterized probability distribution $P(Y|h)$, where
 - ▶ $Y = \{y_1, \dots, y_m\}$ is the full data $y_i = x_i \cup z_i$
 - ▶ h are the parameters

Determine:

- ▶ h that (locally) maximizes $E[\ln P(Y|h)]$

Many uses:

- ▶ Train Bayesian belief networks
- ▶ Unsupervised clustering (e.g., k means)
- ▶ Hidden Markov Models

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.

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