# Bayesian Learning

Berlin Chen 2004

References:

1. Machine Learning , Chapter 6
2. Tom M. Mitchell’s teaching materials
3. Artificial Intelligence: A Modern Approach, Chapter 14
4. Russell’s teaching materials

# Bayes Theorem

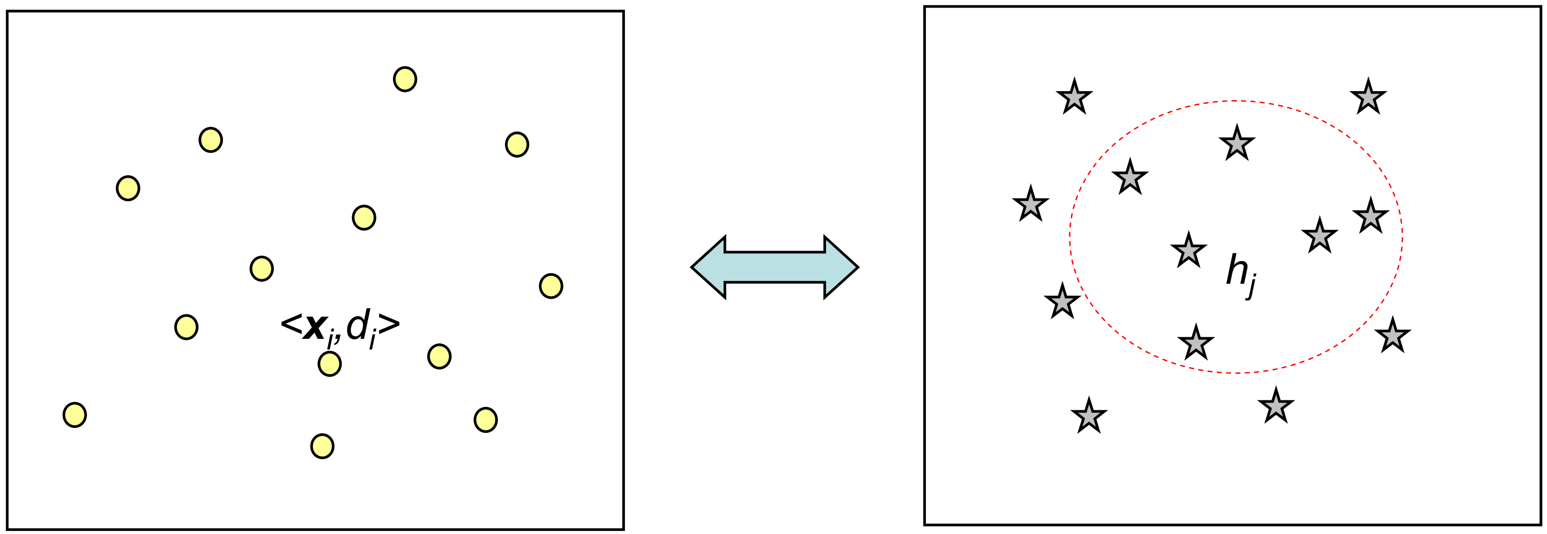
$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- $P(h)$ : 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*

# Bayes Theorem

- Related to machine learning problems

Instance Space *D*　　　　Hypothesis Space *H*




A set of training data *D*

A set of hypotheses (candidate target functions)

$h_j(x_i) =?\ d_i$

playing tennis
not playing tennis

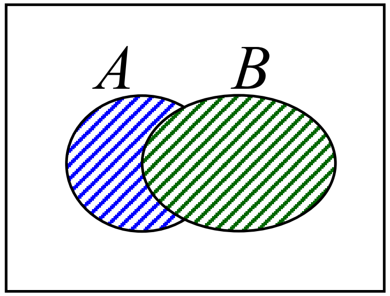
# Review: Basic Formulas for Probabilities

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

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

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

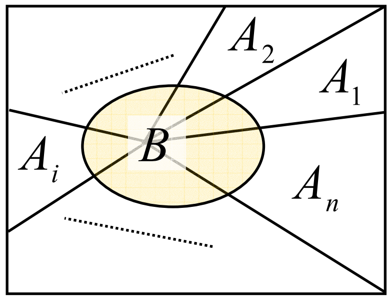
$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$




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

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

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



# Choosing Hypotheses: *MAP* Criterion

- In machine learning, we are interested in finding the best (most probable) hypothesis *h* from some hypothesis space *H*, given the observed (training) data *D*

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

  – A Maximum a Posteriori (*MAP*) hypothesis $h_{MAP}$

# Choosing Hypotheses: *ML* Criterion

- If we further assume that every hypothesis is equally probable a priori, e.g. $P(h_i) = P(h_j)$. The above equation can be simplified as:

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

  – A Maximum Likelihood (*ML*) hypothesis $h_{ML}$

- $P(D|h)$ often called "the likelihood of the data *D* given *h*"

# Example

$P(Cancer|+)$ ?

$P(\neg Cancer|+)$ ?

- Does patient have cancer or not ?
  - A patient takes a lab test
    1. The result comes back positive
    2. 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

$$P(+|Cancer) = 0.98, \qquad P(-|Cancer) = 0.02$$

$$P(+|\neg Cancer) = 0.03, \qquad P(-|\neg Cancer) = 0.97$$

$$P(Cancer) = 0.008, \qquad P(\neg Cancer) = 0.992$$

$$P(+|Cancer)P(Cancer) = 0.98 \times 0.008 = 0.0078$$

$$P(+|\neg Cancer)P(\neg Cancer) = 0.03 \times 0.992 = 0.298$$

$$P(Cancer|+) = \frac{0.0078}{0.0078 + 0.298} = 0.21$$

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

# Brute Force MAP Learning Algorithm

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

$$h_{MAP} = \underset{h \in H}{\arg\max}\, P(h|D)$$

# Relation to Concept Learning

- Consider the concept learning task
  - Instance space *X*, hypothesis space *H*, training examples *D*
  - Consider FINDS learning algorithm
    - Output the most specific hypothesis from the Version Space $VS_{H,D}$

- Does FINDS output a MAP hypothesis ?

# Bayesian Analysis for Concept Learning

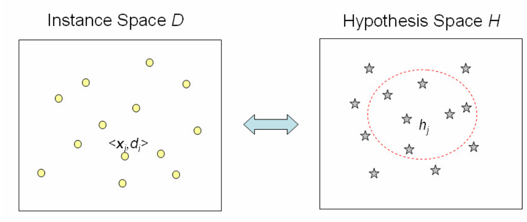
- Assume a fixed set of instances $<x_1, x_2, \ldots, x_m>$, and D is the set of target values (classifications) for the above fixed set of instances
  $D=<d_1, d_2, \ldots, d_m>$

assumption:
noise-free training data

- Choose P($D$|$h$)
  - $P(D|h)=1$ if $h$ is consistent with $D$
    - $h$ is in the Version Space
  - $P(D|h)=0$ otherwise

deterministic prediction

- Choose P($h$) to be uniform (prior) distribution (?)
  - $P(h)=\frac{1}{|H|}$ for all $h$ in $H$
- Then, $P(h|D)=\begin{cases} \frac{1}{|VS_{H,D}|} & \text{if } h \text{ is consistent with } D \\ 0 & \text{otherwise} \end{cases}$



# Bayesian Analysis for Concept Learning

- Explanation

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

$$= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} + \sum_{h_i \notin VS_{H,D}} 0 \cdot \frac{1}{|H|}$$

$$= \frac{|VS_{H,D}|}{|H|}$$

Suppose that hypotheses are mutually exclusive

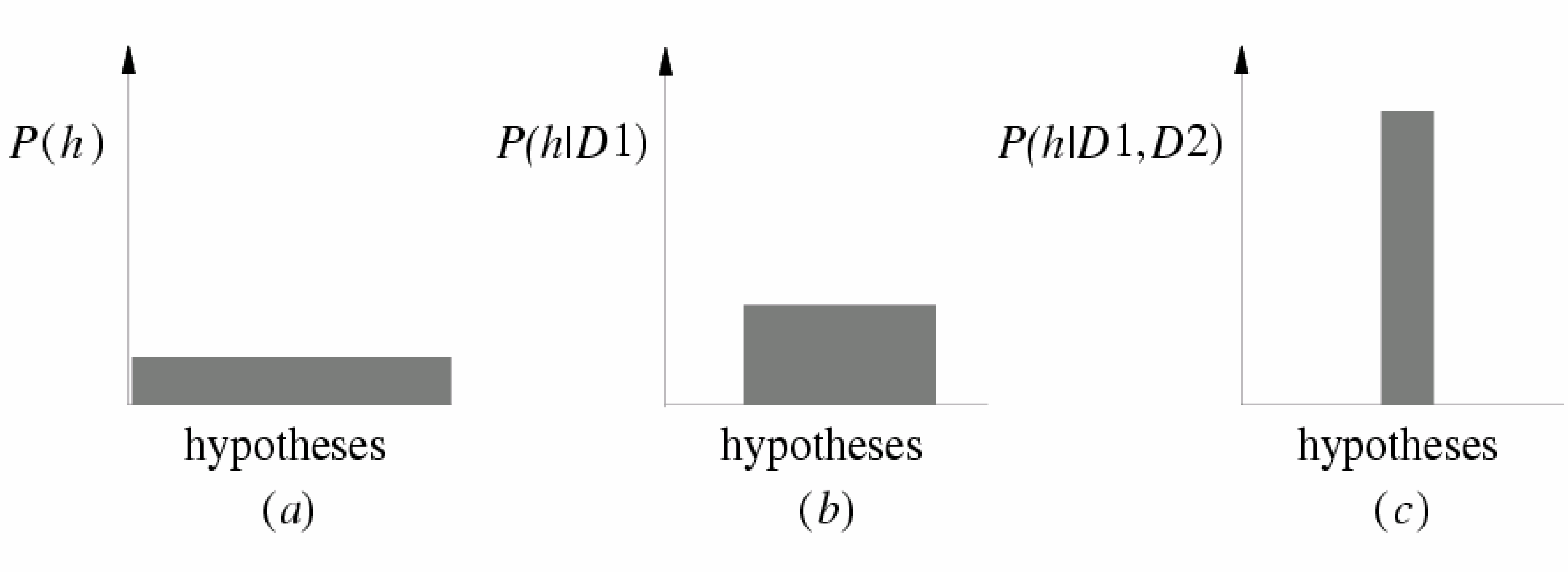
$$P(h|D) = \frac{P(D|h)P(h)}{P(D)} = \frac{P(D|h)\frac{1}{|H|}}{\frac{|VS_{H,D}|}{|H|}} = \frac{P(D|h)}{|VS_{H,D}|}$$

– if *h* is consistent with *D*, i.e. *P*(*D*|*h*)=1 ⇨ $P(h|D) = \frac{1}{|VS_{H,D}|}$

– if *h* is inconsistent with *D*, i.e. *P*(*D*|*h*)=0 ⇨ $P(h|D) = 0$

# Bayesian Analysis for Concept Learning

- Evolution of posterior probabilities $P(h|D)$ with increasing training data



# Bayesian Analysis for Concept Learning

- For FINDS that favors the most specific hypothesis, the prior distribution $P(h)$ over *H* can assign

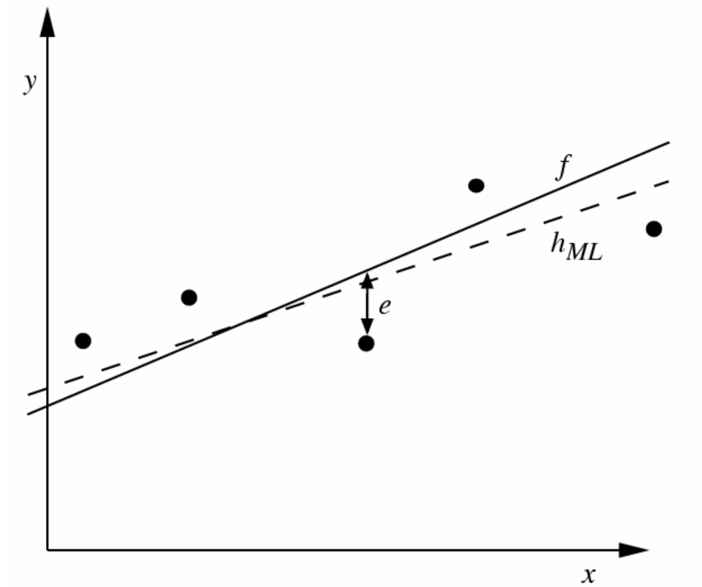
$$P(h_1) \geq P(h_2) \text{ if } h_1 \text{ is more specific than } h_2$$

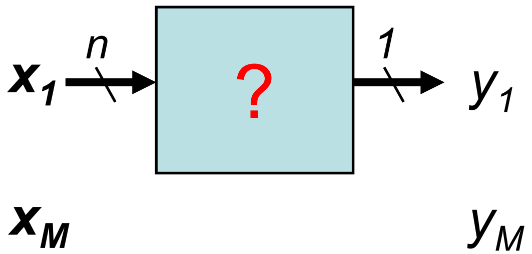
  - Then, FINDS output a MAP hypothesis

The Bayesian framework provide a way to characterize the behavior of learning algorithms, even the algorithm does not explicitly manipulate probabilities.

# Learning A Real-Valued Target Function

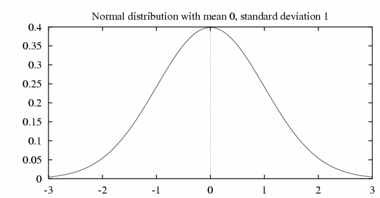
- Applications: neural network, linear regression, curve fitting
  - Minimize the sum of squared errors between the output hypothesis predictions and the training data




  - The Bayesian analysis will show that the hypothesis obtained is just a Maximum Likelihood (ML) hypothesis under certain assumptions

# Learning A Real-Valued Target Function




- Consider any real-valued target function $f$
  - Training examples <$x_i$, $d_i$>, where $d_i$ is corrupted by random noise
    desired target 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, variance= $\sigma^2$)
    - Therefore, each $d_i$ also forms a Gaussian distribution (with mean= $f(x_i)$, variance= $\sigma^2$)

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

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

# Learning A Real-Valued Target Function

Assume training examples are mutually independent given hypothesis *h*

$$h_{ML} = \arg\max_{h \in H} p(D|h)$$

$$= \arg\max_{h \in H} \prod_{i=1}^{m} p(d_i|h)$$

$p(d_i|h)$ can be expressed as a Normal distribution

$$= \arg\max_{h \in H} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{d_i - h(x_i)}{\sigma}\right)^2}$$

$h(x_i) \rightarrow f(x_i)$

- Instead, maximize the natural logarithm of above equation

$$h_{ML} = \arg\max_{h \in H} \ln p(D|h)$$

$$= \arg\max_{h \in H} \ln \sum_{i=1}^{n} \ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2}\left(\frac{d_i - h(x_i)}{\sigma}\right)^2$$

$$= \arg\max_{h \in H} \ln \sum_{i=1}^{n} -\frac{1}{2}\left(\frac{d_i - h(x_i)}{\sigma}\right)^2$$

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

# Learning A Real-Valued Target Function

- Assumptions we have made
  - Noise occurs in the target values $d_i$
    - The training values are generated by adding random noise to the target value, where the random noise is drawn independently for each example from a Normal distribution with zero mean
  - Do not consider noise in the attributes $x_i$

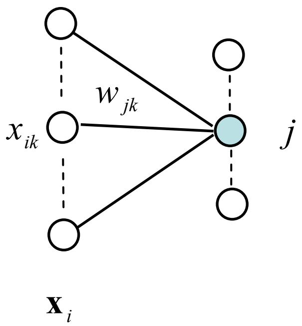
# Learning To Predict Probabilities

- Consider predicting survival probability from patient data
  - Training examples <$x_i$, $d_i$>, where $d_i$ is 1 (survival) or 0 (death)

$$f : X \rightarrow \{0, 1\} \quad f(x) = 0 \text{ or } 1$$

- Want to train a neural network to output a probability of survival given $x_i$ (not 0 and 1)

$$f' : X \rightarrow [0, 1], \quad f'(x) = P(f(x) = 1)$$




$$P(D|h) = \prod_{i=1}^{m} P(x_i, d_i|h)$$

each training example is drawn independent

$$= \prod_{i=1}^{m} P(d_i|h, x_i)P(x_i)$$

The occurrence of $x_i$ is independent of $h$ a ($x_i$ is treated as a random variable)

$$P(d_i|h, x_i) = \begin{cases} h(x_i) & \text{if } d_i = 1 \\ 1 - h(x_i) & \text{if } d_i = 0 \end{cases}$$

$$\Rightarrow P(d_i|h, x_i) = h(x_i)^{d_i}(1 - h(x_i))^{1-d_i}$$

# Learning To Predict Probabilities

$$P(D|h) = \prod_{i=1}^{m} h(x_i)^{d_i} (1 - h(x_i))^{1-d_i} P(x_i)$$

- The maximum likelihood hypothesis

$$h_{ML} = \arg\max_{h \in H} \prod_{i=1}^{m} h(x_i)^{d_i} (1 - h(x_i))^{1-d_i} \boxed{P(x_i)}$$

dropped !

$$= \arg\max_{h \in H} \prod_{i=1}^{m} h(x_i)^{d_i} (1 - h(x_i))^{1-d_i}$$

- The logarithm of the likelihood

$G(h, D)$

$$h_{ML} = \arg\max_{h \in H} \boxed{\sum_{i=1}^{m} d_i \ln h(x_i) + (1 - d_i) \ln h(1 - h(x_i))}$$

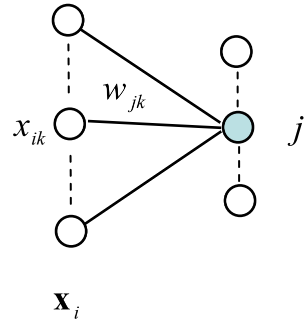
$$= \boxed{\arg\min_{h \in H} \sum_{i=1}^{m} - d_i \ln h(x_i) - (1 - d_i) \ln(1 - h(x_i))}$$

Minimize the cross entropy

# Learning To Predict Probabilities

- Suppose the neural network is a single layer of sigmoid units
  - Sigmoid function

$$h(\mathbf{x}_i) = \frac{1}{1+\exp\left(-\mathbf{w}_j \cdot \mathbf{x}_i\right)} = \frac{1}{1+\exp\left(-\left(\sum_{k=1}^{K} w_{jk} x_{ik}\right)\right)}$$




- Perform gradient-ascent search for *h*
  - Take partial derivate of $G(h,D)$ with respect to $w_{jk}$

$$\frac{\partial G(h,D)}{\partial w_{jk}} = \sum_{i=1}^{m} \frac{\partial\left(d_i \ln h(\mathbf{x}_i) + (1-d_i)\ln\left(1-h(\mathbf{x}_i)\right)\right)}{\partial h(\mathbf{x}_i)} \frac{\partial h(\mathbf{x}_i)}{\partial w_{jk}}$$

$$= \sum_{i=1}^{m} \left[\frac{d_i - h(\mathbf{x}_i)}{h(\mathbf{x}_i)\left(1-h(\mathbf{x}_i)\right)}\right] \cdot \left[h(\mathbf{x}_i)\left(1-h(\mathbf{x}_i)\right)x_{ik}\right]$$

$$= \sum_{i=1}^{m} \left(d_i - h(\mathbf{x}_i)\right)x_{ik}$$

# Learning To Predict Probabilities

- Explanation

$$\frac{\partial h(\mathbf{x}_i)}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}}\left[\frac{1}{1+\exp\left(-\sum_{k=1}^{K} w_{jk} x_{ik}\right)}\right]$$

$$= \frac{x_{ik}\exp\left(-\sum_{k=1}^{K} w_{jk} x_{ik}\right)}{\left[1+\exp\left(-\sum_{k=1}^{K} w_{jk} x_{ik}\right)\right]^2}$$

$$= x_{ik}\cdot h(\mathbf{x}_i)\frac{\exp\left(-\sum_{k=1}^{K} w_{jk} x_{ik}\right)}{1+\exp\left(-\sum_{k=1}^{K} w_{jk} x_{ik}\right)}$$

$$= x_{ik}\cdot h(\mathbf{x}_i)\left(1-h(\mathbf{x}_i)\right)$$

# Learning To Predict Probabilities

- On each iteration of the search, the weight vector is adjusted in the direction of the gradient
  - Gradient Ascent Search (a kind of local search)

$$\hat{w}_{jk} \leftarrow w_{jk} + \Delta w_{jk}$$

where

$$\Delta w_{jk} = \eta \sum_{i=1}^{m} \left(d_i - h(\boldsymbol{x}_i)\right) x_{ik}$$

  - $\eta$ is a small positive constant that determines the step size of the gradient search

# Minimum Description Length Principle

- Occam's razor: prefer the shortest hypothesis
- A derivation for the MAP hypothesis

$$h_{MAP} = \arg\max_{h \in H} P(D|h)P(h)$$

think of it as the minimum number of bits (description length) needed to encode *h* using an optimal encoder

  - Take the $\log_2$ operation

$$h_{MAP} = \arg\max_{h \in H} \log_2 P(D|h) + \log_2 P(h)$$

  - Take the negation operation

$$h_{MAP} = \arg\min_{h \in H} -\log_2 P(D|h) - \log_2 P(h)$$

    - Interpreted as a statement that short hypotheses are preferred, under an optimal encoding of hypothesis and data

# Minimum Description Length Principle

- The MAP hypothesis minimizes the sum given by the description length of the hypothesis plus the description length of the data given the hypothesis

$$h_{MAP} = \arg\max_{h \in H} L_{C_H}(h) + L_{C_{D|h}}(D|h)$$

where

$$L_{C_H}(h) = -\log_2 P(h)$$

$$L_{C_{D|h}}(D|h) = -\log_2 P(D|h)$$

# Most Probable Classification of New Instances

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

$$h_{MAP} = \arg\max_{h \in H} P(h|D)$$

- Given new instance *x* what is its most probable classification?
  - $h_{MAP}(x)$ is not the most probable classification !
  - The most probable classification is obtained by combining the predictions of all hypotheses

# Most Probable Classification of New Instances

- Example
  - Three hypotheses:

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

  - Given new instance $x$

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

deterministic prediction

  - What is the most probable classification of $x$ ?
    - Only $h_1$ is considered, $x$ is classified positive
    - If all hypotheses are considered,
      $x$ is classified positive by $h_1$ with probability 0.4
      $x$ is classified negative by $h_2$ and $h_3$ with probability 0.6

# Bayes Optimal Classifier

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

- Bayes Optimal Classification

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

- For the previous example

$$P(h_1|D) = 0.4,\ P(+|h_1) = 1.0,\ P(-|h_1) = 0.0$$
$$P(h_2|D) = 0.3,\ P(+|h_2) = 0.0,\ P(-|h_2) = 1.0$$
$$P(h_3|D) = 0.3,\ P(+|h_3) = 0.0,\ P(-|h_3) = 1.0$$

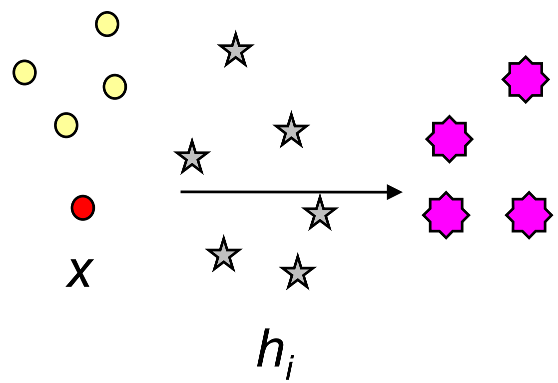
  - Therefore

$v_1$: $$\sum_{h_i \in H} P(+|h_i)P(h_i|D) = 0.4$$

$v_2$: $$\sum_{h_i \in H} P(-|h_i)P(h_i|D) = 0.6$$

  - And

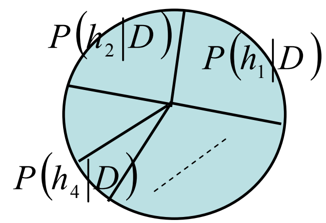
$$\arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D) = -$$



# 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:

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

  – Haussler et al. 1994

# Gibbs Classifier

- Suppose the above expectation of error for Gibbs classifier is correct.
  Then consider the concept learning using version space with uniform prior distribution over *H*
  - Pick any hypothesis from VS with uniform probability
  - Its expected error no worse than twice that of the Bayes optimal classifier

# Naïve 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

$$v_{MAP} = \arg\max_{v_j \in V} P(v_j | a_1, a_2, \ldots, a_n)$$

predict the target value/classification

$$= \arg\max_{v_j \in V} \frac{P(a_1, a_2, \ldots, a_n | v_j) P(v_j)}{P(a_1, a_2, \ldots, a_n)}$$

$$= \arg\max_{v_j \in V} P(a_1, a_2, \ldots, a_n | v_j) P(v_j)$$

  - Naïve Bayes assumption: $P(a_1, a_2, \ldots, a_n | v_j) = \prod_i P(a_i | v_j)$

  - Naïve Bayes Classifier: $v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$

# Naïve Bayes: Example 1

- Given a data set **Z** with 3-dimensional Boolean examples. Train a naïve Bayes classifier to predict the classification

| Attribute A | Attribute B | Attribute C | Classification D |
|---|---|---|---|
| F | T | F | T |
| F | F | T | T |
| T | F | F | T |
| T | F | F | F |
| F | T | T | F |
| F | F | T | F |

$P(D=T)=1/2,\ P(D=F)=1/2$

$P(A=T|D=T)=1/3, P(A=F|D=T)=2/3$
$P(B=T|D=T)=1/3, P(B=F|D=T)=2/3$
$P(C=T|D=T)=1/3, P(B=F|D=T)=2/3$

$P(A=T|D=F)=1/3, P(A=F|D=F)=2/3$
$P(B=T|D=F)=1/3, P(B=F|D=F)=2/3$
$P(C=T|D=F)=2/3, P(B=F|D=F)=1/3$

  - What is the predicted probability $P(D=T|A=T,B=F,C=T)$ ?
  - What is the predicted probability $P(D=T|B=T)$ ?

# Naïve Bayes: Example 1

$$P(D=T \mid A=T, B=F, C=T)$$
$$= \frac{P(A=T, B=F, C=T \mid D=T)P(D=T)}{P(A=T, B=F, C=T)}$$
$$= \frac{P(A=T, B=F, C=T \mid D=T)P(D=T)}{P(A=T, B=F, C=T \mid D=T)P(D=T) + P(A=T, B=F, C=T \mid D=F)P(D=F)}$$
$$= \frac{\frac{1}{3}\cdot\frac{2}{3}\cdot\frac{1}{3}\cdot\frac{1}{2}}{\frac{1}{3}\cdot\frac{2}{3}\cdot\frac{1}{3}\cdot\frac{1}{2} + \frac{1}{3}\cdot\frac{2}{3}\cdot\frac{2}{3}\cdot\frac{1}{2}} = \frac{2}{2+4} = \frac{1}{3}$$

$$P(D=T \mid B=T)$$
$$= \frac{P(B=T \mid D=T)P(D=T)}{P(B=T)}$$
$$= \frac{P(B=T \mid D=T)P(D=T)}{P(B=T \mid D=T)P(D=T) + P(B=T \mid D=F)P(D=F)}$$
$$= \frac{\frac{1}{3}\cdot\frac{1}{2}}{\frac{1}{3}\cdot\frac{1}{2} + \frac{1}{3}\cdot\frac{1}{2}} = \frac{1}{1+1} = \frac{1}{2}$$

# How to Train a Naïve Bayes Classifier

- Naïve_Bayes_Learn(*examples*)

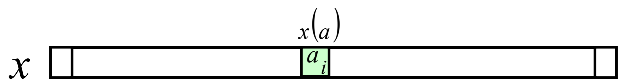
For each target value $v_j$

$$\hat{P}(v_j) \leftarrow \text{maximum likelihood (ML) estimate of } P(v_j)$$

$$\frac{|v_j|}{n} \; or \; \frac{|v_j|}{\sum_{v_k} |v_k|}$$

For each attribute value $a_i$ of each attribute $a$

$$\hat{P}(a_i | v_j) \leftarrow \text{maximum likelihood (ML) estimate of } P(a_i | v_j)$$




$$\frac{\sum_{x \in v_j, x(a)=a_i} 1}{\sum_{x \in v_j} 1} = \frac{\sum_{x \in v_j, x(a)=a_i} 1}{|v_j|}$$

- Classify_New_Instance($x$)

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_{a_i \in x} P(a_i | v_j)$$

# Naïve Bayes: Example 2

- Consider *PlayTennis* again and new instance
  <*Outlook*=*sunny*, *Temperature*=*cool*, *Humidity*=*high*, *Wind*=*strong*>

- Want to compute

$$v_{NB} = \arg \max_{v_j \in V = \{yes, no\}} P(v_j) \times P(Outlook = sunny | v_j) \times P(Temperature = cool | v_j) \times P(Humidity = high | v_j) \times P(Wind = Strong | v_j)$$

$$P(yes) \times P(Outlook = sunny | yes) \times P(Temperature = cool | yes) \times P(Humidity = high | yes) \times P(Wind = Strong | yes) = 0.0053$$

$$P(no) \times P(Outlook = sunny | no) \times P(Temperature = cool | no) \times P(Humidity = high | no) \times P(Wind = Strong | no) = 0.206$$

$$\therefore v_{NB} = no$$

# Dealing with Data Sparseness

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

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

$$v_{NB} = \arg\max_{v_j \in V} \hat{P}(v_j) \prod_i \hat{P}(a_i | v_j)$$

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

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

Smoothing

  - $n$ is number of training examples for which $v = v_j$
  - $n_c$ is number of training 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)

# Example: Learning to Classify Text

- For instance,
  - Learn which news articles are of interest
  - Learn to classify web pages by topic
- Naïve Bayes is among most effective algorithms
- What attributes shall we use to represent text documents
  - The word occurs in each document position

# Example: 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
     - $P(+)$
     - $P(-)$
     - $P(doc|+)$
     - $P(doc|-)$
- Naïve 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: $P(a_i = w_k | v_j) = P(a_m = w_k | v_j), \forall\, i, m$

Time Invariant

# Example: Learning to Classify Text

- Learn_Naïve_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
   - $docs_j \leftarrow$ 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
     - $P(w_k | v_j) \leftarrow \frac{n_k + 1}{n + |Vocabulary|}$

Smoothed unigram

# Example: Learning to Classify Text

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

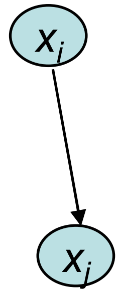
$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_{i \in positions} P(a_i | v_j)$$

# Bayesian Belief Networks

- Premise
  - Naïve Bayes assumption of conditional independence too restrictive
  - But it is 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
- Bayesian Belief Networks also called
  - Bayesian Networks, Bayes Nets, Belief Networks, Probabilistic Networks, etc.

# Bayesian Belief Networks

- A simple, graphical notation for conditional independence assertions and hence for compact specification of full joint distributions

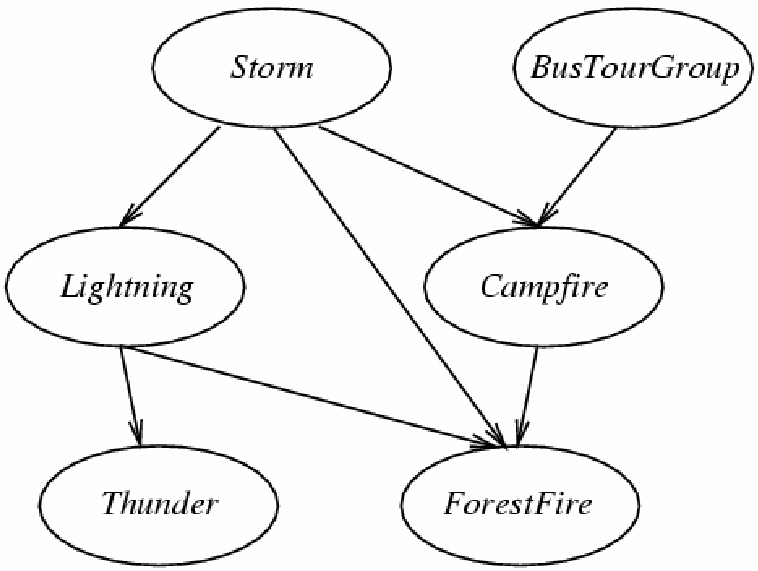



- Syntax
  - A set of nodes, one per variable (discrete or continuous)
  - A directed, acyclic graph (link/arrow ≈ "directly influences")
  - A conditional distribution for each node given its parents

$$P\left(X_i \middle| Parents\left(X_i\right)\right)$$

- In the simplest case, conditional distribution represented as a Conditional Probability Table (CPT) giving the distribution over $P\left(X_i\right)$ for each combination of parent values

# Bayesian Belief Networks




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


| S | B | P(C) |
|---|---|---|
| T | T | 0.4 |
| T | F | 0.1 |
| F | T | 0.8 |
| F | F | 0.2 |

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

# 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 can write

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

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

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

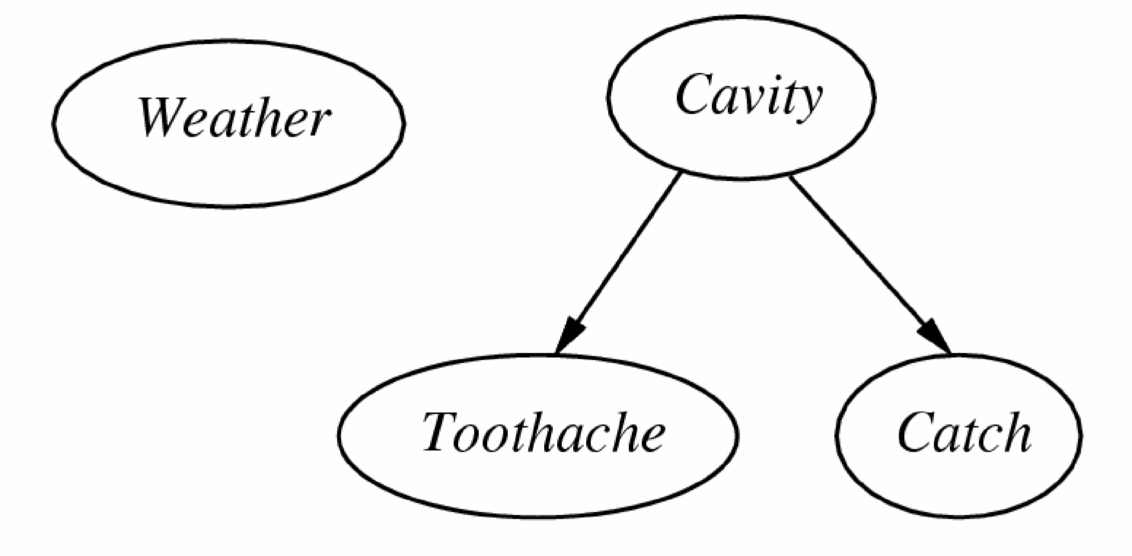
  – Naïve Bayes uses conditional independence to justify

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

X,Y are mutually independent given Z

# Example 1:Dentist Network

- Topology of network encodes conditional independence assertions




  - Weather is independent of the other variables
  - Toothache and Catch are conditionally independent given Cavity
    - Cavity is a direct cause of Toothache and Catch

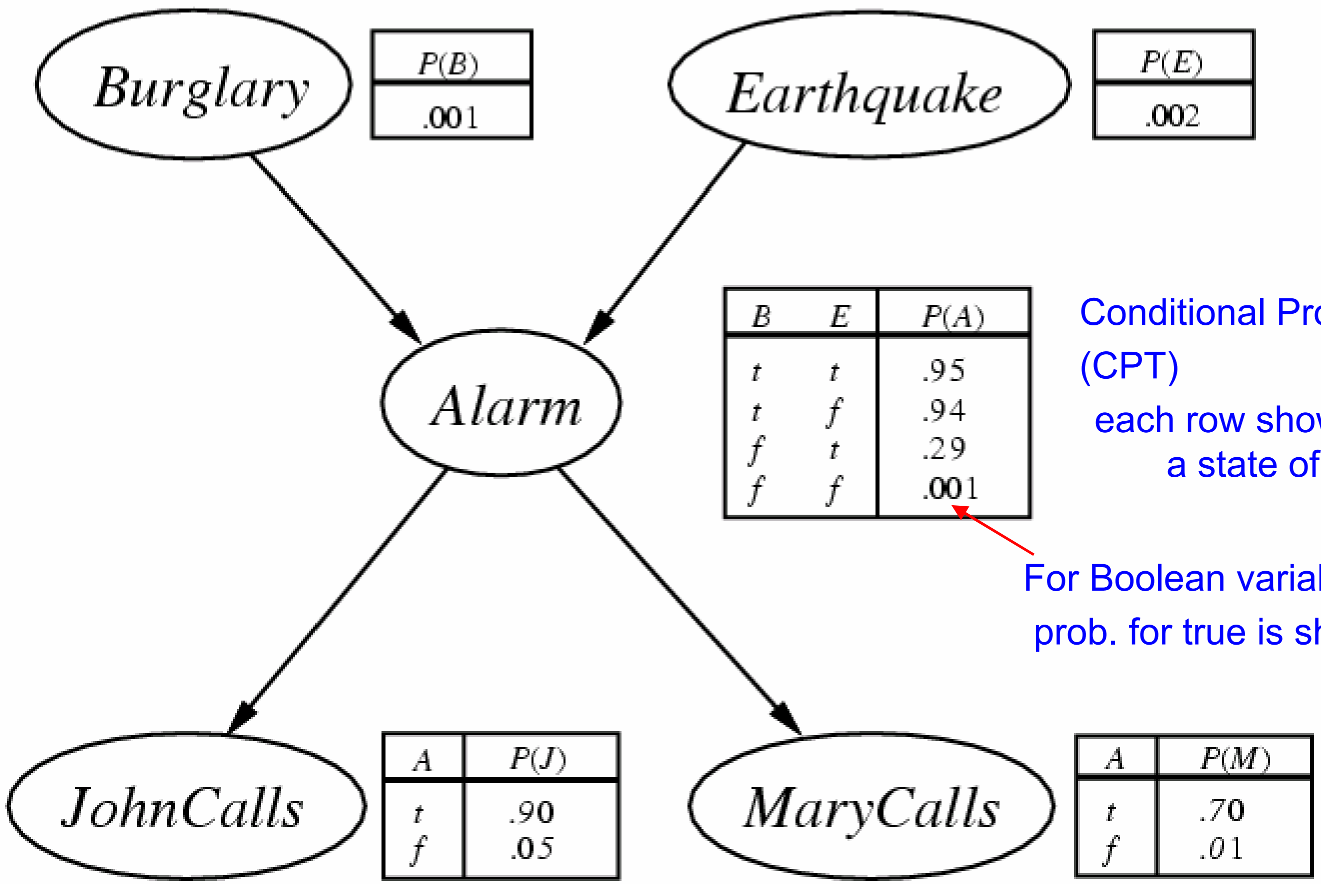
# Example 2: Burglary Network

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

$$P\left(Burglary = T \middle| JohnCall = T, MaryCall = F\right)?$$

  - 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
  - But
    - John sometimes confuses the telephone ringing with the alarm
    - Mary likes rather loud music and sometimes misses the alarm

# Example 2: Burglary Network




| P(B) |
|---|
| .001 |

| P(E) |
|---|
| .002 |

| B | E | P(A) |
|---|---|---|
| t | t | .95 |
| t | f | .94 |
| f | t | .29 |
| f | f | .001 |

| A | P(J) |
|---|---|
| t | .90 |
| f | .05 |

| A | P(M) |
|---|---|
| t | .70 |
| f | .01 |

Conditional Probability Table (CPT)

each row shows prob. given a state of parents

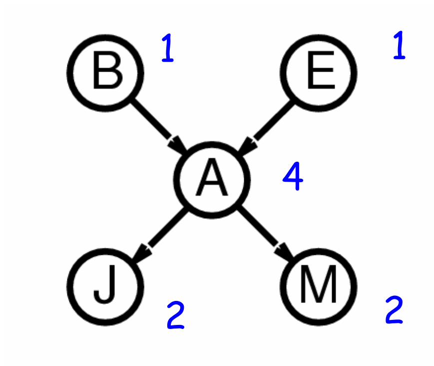
For Boolean variables, just the prob. for true is shown

# 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 $X_i$ = true (the number for $X_i$ = 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)

Chain rule

$$P(B,E,A,J,M)$$
$$= \overset{1}{P(B)}\,\overset{2}{P(E|B)}\,\overset{4}{P(A|B,E)}\,\overset{8}{P(J|B,E,A)}\,\overset{16}{P(M|B,E,A,J)}$$
$$\approx P(B)P(E)P(A|B,E)P(J|A)P(M|A)$$

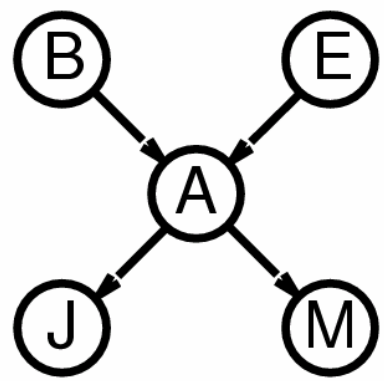


# Global Semantics

- Global semantics defines the full joint distribution as the product of the local conditional distributions

$$P(X_1,\ldots,X_n) \approx \prod_{i=1}^{n} P(X_i \mid Parents(X_i))$$

  - The Bayesian Network is semantically
    - A representation of the joint distribution
    - A encoding of a collection of conditional independence statements




- E.g.,

$$P(J \wedge M \wedge A \wedge \neg B \wedge \neg E)$$

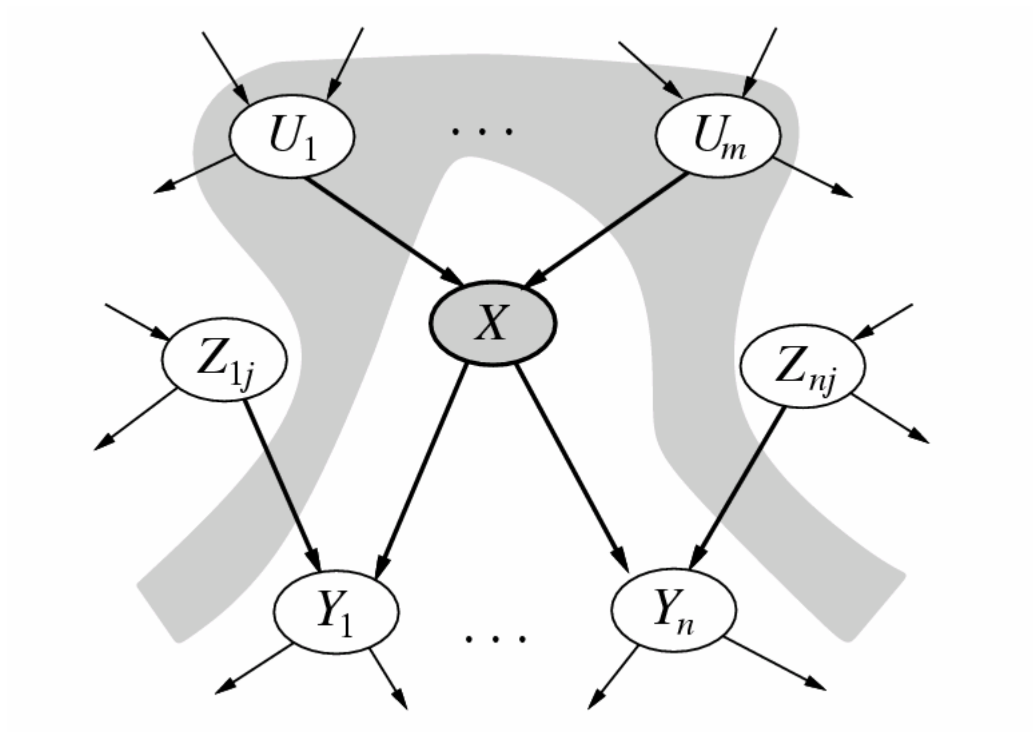
$$\approx P(J \mid A)P(M \mid A)P(A \mid \neg B \wedge \neg E)P(\neg B)P(\neg E)$$

$$= 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998$$

$$= 0.00062$$

# Local Semantics

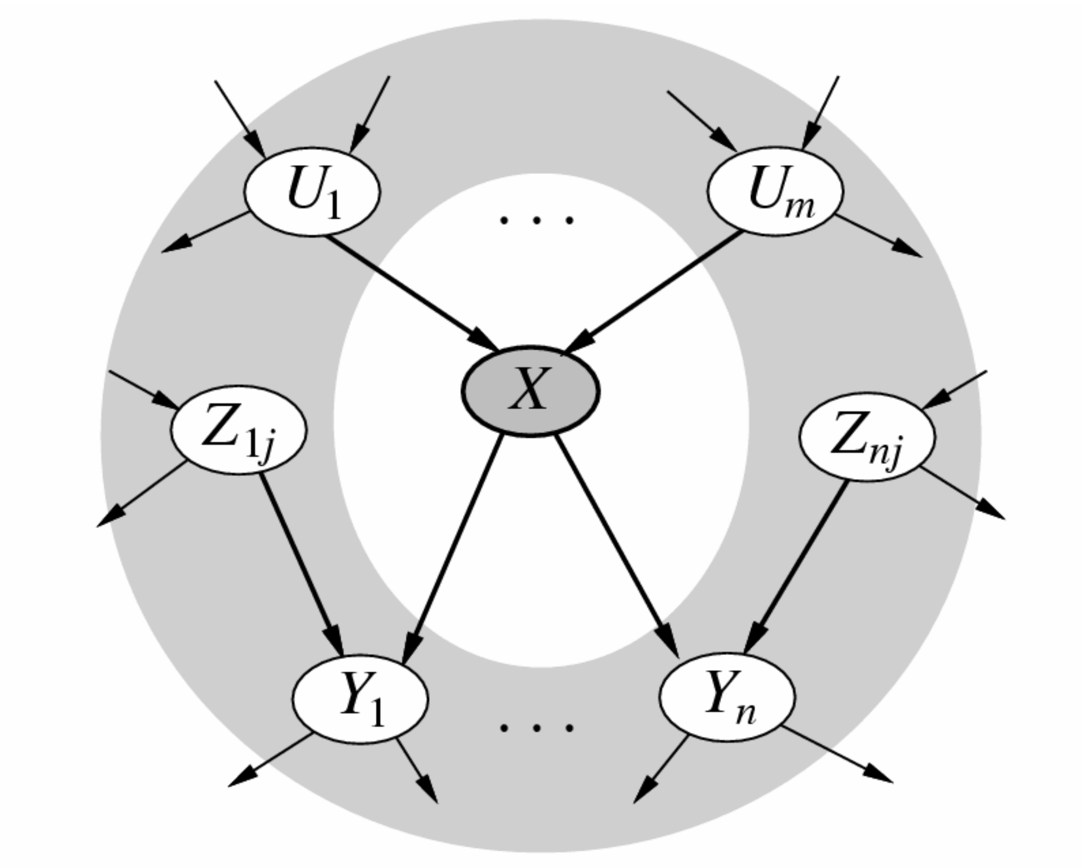
- Local semantics: each node is conditionally independent of its nondescendants given its parents




  – Local semantics ⟺ global semantics

# Markov Blanket

- Each node is conditionally independent of all others given its parents + children + children's parents



# Constructing Bayesian Networks

- Need a method such that a series of locally testable assertions of conditional independence guarantees the required global semantics

  1. Choose an ordering of variables $X_1,..X_i..,X_n$
  2. For *i*=1 to *n*

     add $X_i$ to the network and select parents from $X_1,..X_{i-1}$ such that

     $Parents(X_i) \subseteq \{X_1,..X_{i-1}\}$

     $$P(X_i | X_1,..X_{i-1}) = P(X_i | Parents(X_i))$$

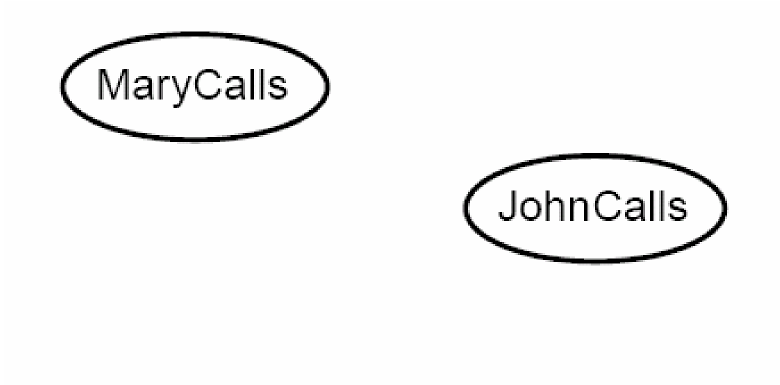
  This choice of parents guarantees the global semantics

$$P(X_1,...,X_n) = \prod_{i=1}^{n} P(X_i | X_1,..X_{i-1}) \qquad \text{(chain rule)}$$

$$= \prod_{i=1}^{n} P(X_i | Parents(X_i)) \qquad \text{(by construction)}$$

# Example for Constructing Bayesian Network

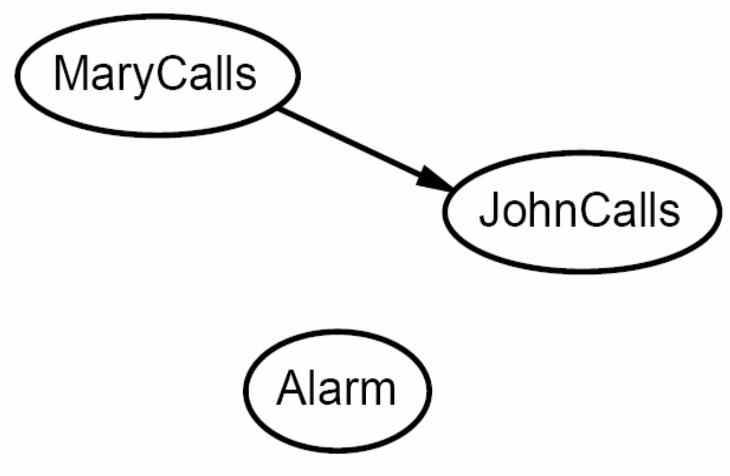
- Suppose we choose the ordering: *M, J, A, B, E*




  – $P(J|M) = P(J)$ ?

# Example for Constructing Bayesian Network

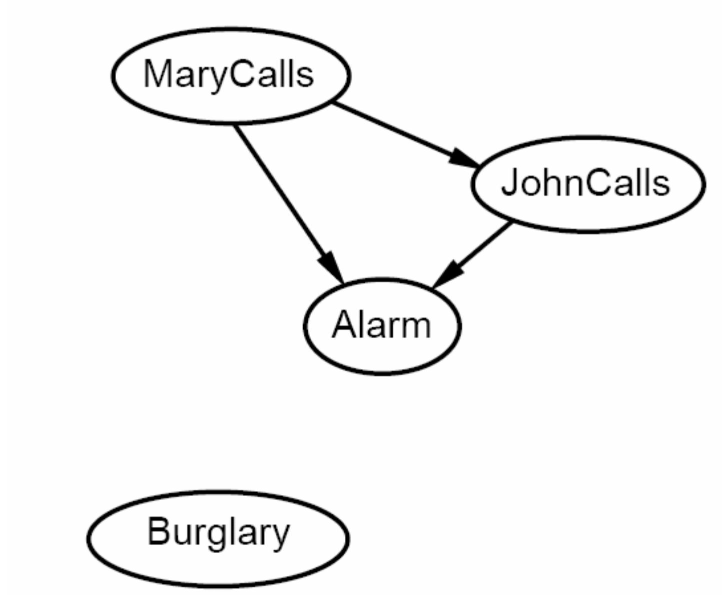
- Suppose we choose the ordering: *M, J, A, B, E*




  - $P(J|M) = P(J)$ ? **No**
  - $P(A|J,M) = P(A|J)$ ? $P(A|J,M) = P(A)$ ?

# Example for Constructing Bayesian Network

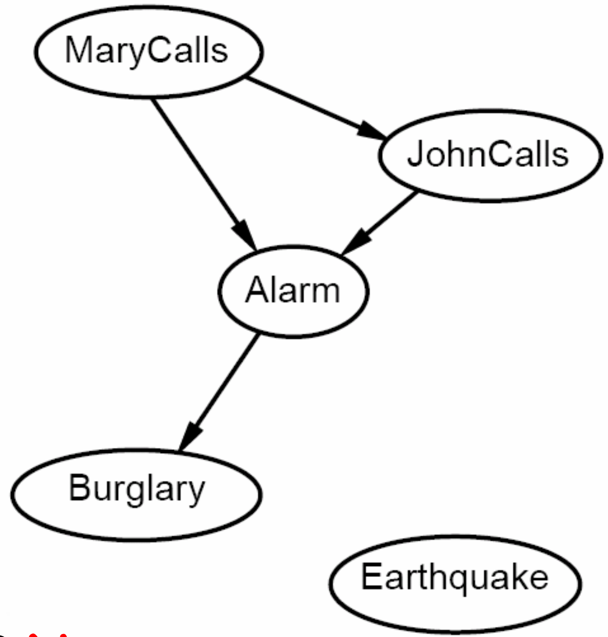
- Suppose we choose the ordering: *M, J, A, B, E*




  - $P(J|M) = P(J)$ ? **No**
  - $P(A|J,M) = P(A|J)$ ? **No** $P(A|J,M) = P(A)$ ? **No**
  - $P(B|A,J,M) = P(B|A)$ ?
  - $P(B|A,J,M) = P(B)$ ?

# Example for Constructing Bayesian Network

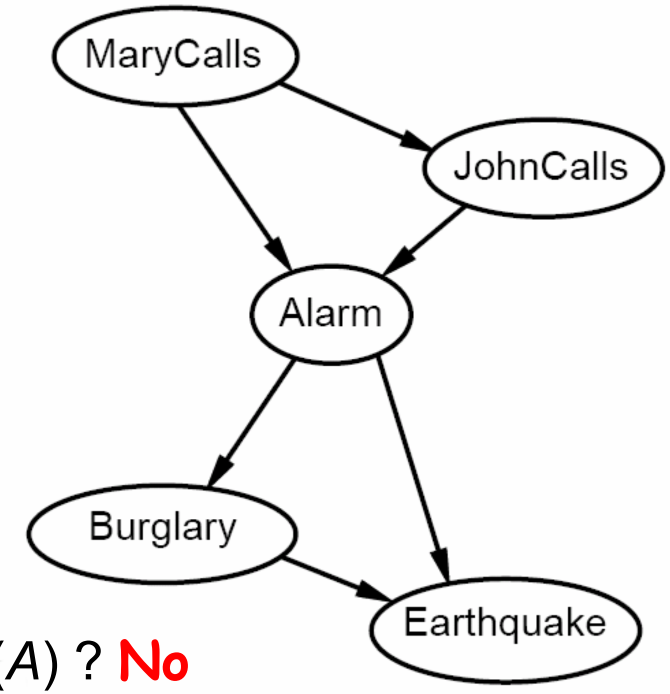
- Suppose we choose the ordering: *M, J, A, B, E*




  - $P(J|M) = P(J)$ ? **No**
  - $P(A|J,M) = P(A|J)$ ? **No** $P(A|J,M) = P(A)$ ? **No**
  - $P(B|A,J,M) = P(B|A)$ ? **Yes**
  - $P(B|A,J,M) = P(B)$ ? **No**
  - $P(E|B,A,J,M) = P(E|A)$ ?
  - $P(E|B,A,J,M) = P(E|B,A)$ ?

# Example for Constructing Bayesian Network

- Suppose we choose the ordering: *M, J, A, B, E*



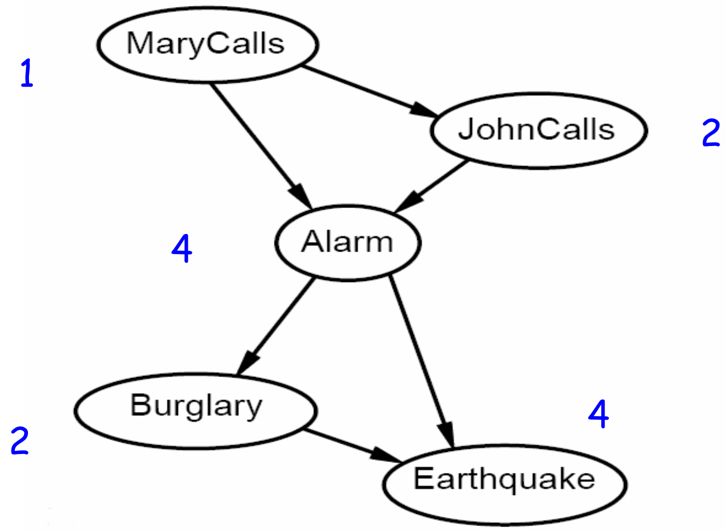

  - $P(J|M) = P(J)$ ? **No**
  - $P(A|J,M) = P(A|J)$ ? **No** $P(A|J,M) = P(A)$ ? **No**
  - $P(B|A,J,M) = P(B|A)$ ? **Yes**
  - $P(B|A,J,M) = P(B)$ ? **No**
  - $P(E|B,A,J,M) = P(E|A)$ ? **No**
  - $P(E|B,A,J,M) = P(E|B,A)$ ? **Yes**

# Example for Constructing Bayesian Network

- Summary
  - Deciding conditional independence is hard in noncausal directions

    (Causal models and conditional independence seem hardwired for humans!)

  - Assessing conditional probabilities is hard in noncausal directions
  - Network is less compact: 1 + 2 + 4 + 2 + 4 = 13 numbers needed



# Inference Tasks

- Simple queries: compute posterior marginal $P(X_i | E = e)$
  - E.g.,

$$P(Burglary | JohnCalls = true, MarryCalls = true)$$

- Conjunctive queries:

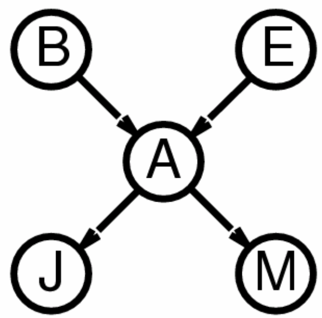
$$P(X_i, X_j | E = e) = P(X_i | X_j, E = e) P(X_j | E = e)$$

- Optimal decisions: probabilistic inference

$$P(Outcome | Action, Evidence)$$

# Inference by Enumeration

- Slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation
- Simple query on the burglary network




$$P(B|j,m) = \frac{P(B,j,m)}{P(j,m)}$$
$$= \alpha P(B,j,m)$$
$$= \alpha \sum_e \sum_a P(B,e,a,j,m)$$

- Rewrite full joint entries using product of CPT entries:

$$P(B|j,m)$$
$$= \alpha \sum_e \sum_a P(B,e,a,j,m)$$
$$= \alpha \sum_e \sum_a P(B)P(e)P(a|B,e)P(j|a)P(m|a)$$
$$= \alpha P(B) \sum_e P(e) \sum_a P(a|B,e)P(j|a)P(m|a)$$

# Evaluation Tree

- Enumeration is inefficient: repeated computation\al
  - e.g., computes $P(j|a)P(m|a)$ for each value of $e$