MODULE-4
BAYESIAN LEARNING

Introduction:

Bayesian learning methods are relevant to our study of machine learning for two different reasons.

Bayesian learning algorithms that calculate explicit probabilities for hypotheses, such as the naive Bayes classifier, are among the most practical approaches to certain types of learning problems.

Bayesian methods are important to our study of machine learning is that they provide a useful perspective for understanding many learning algorithms that do not explicitly manipulate probabilities.

Features of Bayesian learning methods

- Each observed training example can incrementally decrease or increase the estimated probability that a hypothesis is correct. This provides a more flexible approach to learning than algorithms that completely eliminate a hypothesis if it is found to be inconsistent with any single example.
- Prior knowledge can be combined with observed data to determine the final probability of a hypothesis. In Bayesian learning, prior knowledge is provided by asserting

  (1) a prior probability for each candidate hypothesis

  (2) a probability distribution over observed data for each possible hypothesis.
- Bayesian methods can accommodate hypotheses that make probabilistic predictions.
- New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities.
- Even in cases where Bayesian methods prove computationally intractable, they can provide a standard of optimal decision making against which other practical methods can be measured

Difficulty in applying Bayesian methods

- They typically require initial knowledge of many probabilities. When these probabilities are not known in advance they are often estimated based on background knowledge, previously available data, and assumptions about the form of the underlying distributions.
- The computational cost required to determine the Bayes optimal hypothesis in the general case (linear in the number of candidate hypotheses). In certain specialized situations, this computational cost can be significantly reduced.

BAYES THEOREM

In machine learning we are often interested in determining the best hypothesis from some space H, given the observed training data D.

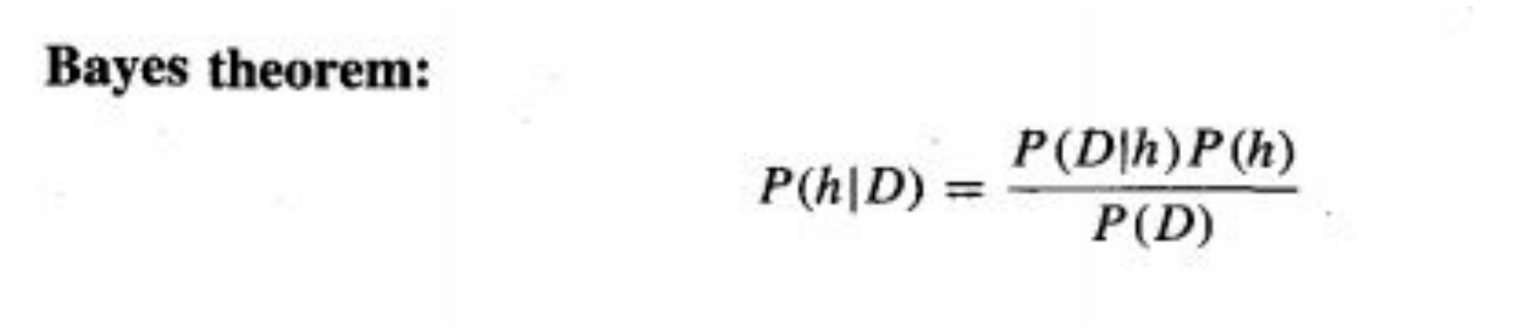
One way to specify the best hypothesis is to say that we demand the most probable hypothesis, given the data D plus any initial knowledge about the prior probabilities of the various hypotheses in H.

Bayes theorem provides a direct method for calculating such probabilities.

Bayes theorem provides a way to calculate the probability of a hypothesis based on its prior probability, the probabilities of observing various data given the hypothesis, and the observed data itself.

BAYES THEOREM

Bayes theorem is the cornerstone of Bayesian learning methods because it provides a way to calculate the posterior probability P(*hlD*), from the prior probability P(h), together with P(D) and P(D/h).



Bayes theorem:

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

P(*hlD*) increases with P(h) and with P(D/h) according to Bayes theorem. It is also reasonable to see that P(*hlD*) decreases as P(D) increases.

MAP hypothesis

- In many learning scenarios, the learner considers some set of candidate hypotheses H and is interested in finding the most probable hypothesis h given the observed data D.
- Any such maximally probable hypothesis is called a **maximum a posteriori (MAP) hypothesis**.
- We can determine the MAP hypotheses by using Bayes theorem to calculate the posterior probability of each candidate hypothesis.
- More precisely, we will say that MAP is a MAP hypothesis provided

$$\begin{aligned} h_{MAP} &\equiv \underset{h\in H}{\operatorname{argmax}}\, P(h|D) \\ &= \underset{h\in H}{\operatorname{argmax}}\, \frac{P(D|h)P(h)}{P(D)} \\ &= \underset{h\in H}{\operatorname{argmax}}\, P(D|h)P(h) \end{aligned}$$

Maximum likelihood (ML) hypothesis

In some cases, we will assume that every hypothesis in H is equally probable a priori.

In this case we can further simplify the above Equation and need only consider the term P(D/h) to find the most probable hypothesis.

P(D/h) is often called the likelihood of the data D given h, and any hypothesis that maximizes P(D/h) is called a **maximum likelihood (ML) hypothesis**, h $_{ML}$

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

An Example

To illustrate Bayes rule, consider a medical diagnosis problem in which there are two alternative hypotheses:

(1) that the patient has a particular form of cancer.

(2) that the patient does not.

The available data is from a particular laboratory test with two possible outcomes: $\oplus$ (positive) and $\ominus$ (negative).

We have prior knowledge that over the entire population of people only .008 have this disease.

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. In other cases, the test returns the opposite result.

An Example

- The above situation can be summarized by the following probabilities:

$$P(cancer) = .008, \qquad P(\neg cancer) = .992$$
$$P(\oplus|cancer) = .98, \qquad P(\ominus|cancer) = .02$$
$$P(\oplus|\neg cancer) = .03, \qquad P(\ominus|\neg cancer) = .97$$

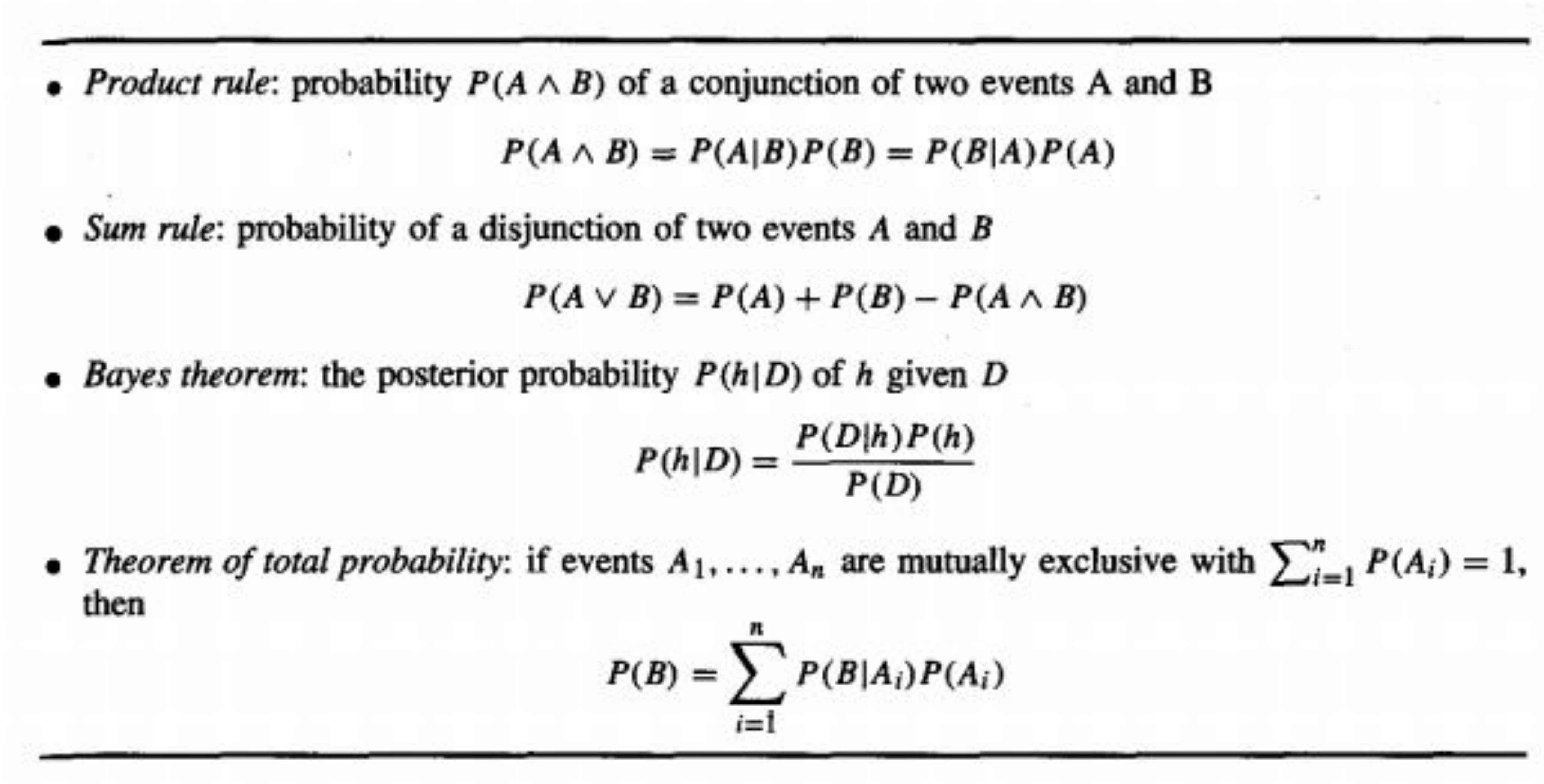
- The maximum a posteriori hypothesis can be found using Equation

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

$$P(\oplus|cancer)P(cancer) = (.98).008 = .0078$$
$$P(\oplus|\neg cancer)P(\neg cancer) = (.03).992 = .0298$$

Thus, $h_{MAP} \equiv \neg cancer.$

Summary of basic probability formulas



- *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

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

- *Bayes theorem*: the posterior probability $P(h|D)$ of h given D

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

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

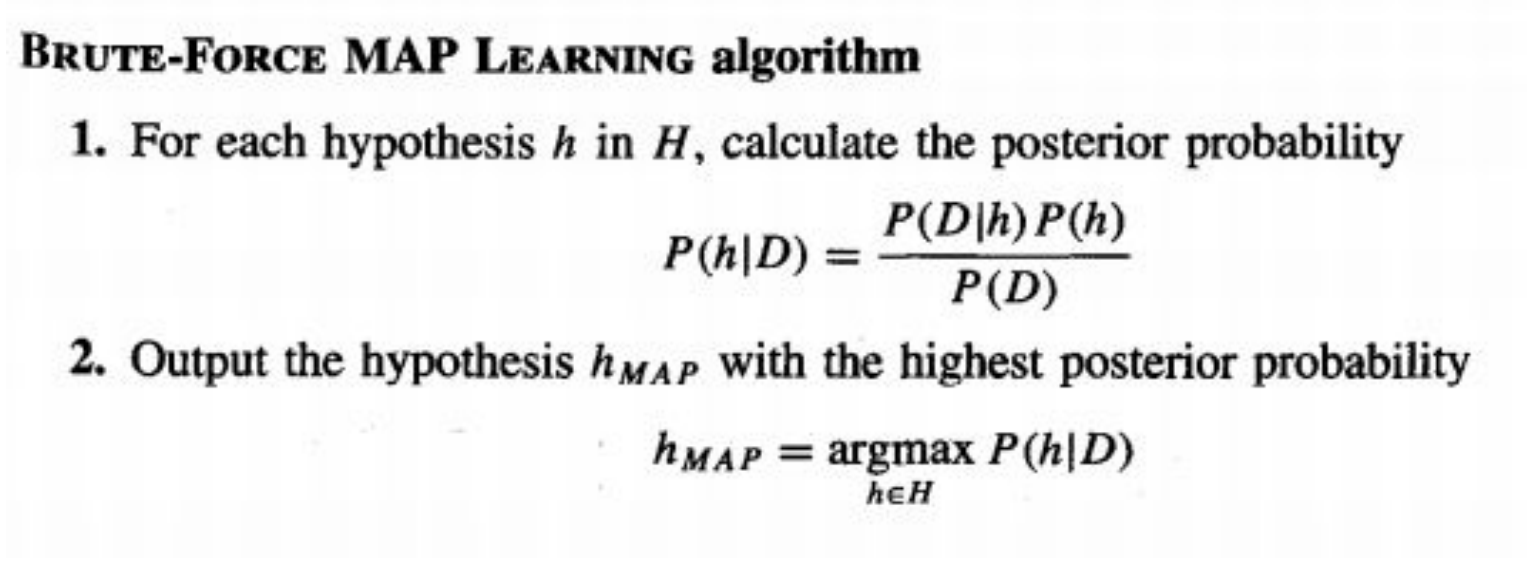
$$P(B) = \sum_{i=1}^{n} P(B|A_i)P(A_i)$$

BAYES THEOREM AND CONCEPT LEARNING

- we can use Bayes theorem as the basis for a straightforward learning algorithm that calculates the probability for each possible hypothesis, then outputs the most probable.

Brute-Force Bayes Concept Learning:

We can design a straightforward concept learning algorithm to output the maximum a posteriori hypothesis, based on Bayes theorem, as follows:



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

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

BRUTE-FORCE MAP LEARNING algorithm

Advantages:

- It provides a standard against which we may judge the performance of other concept learning algorithms.

Disadvantages:

- This algorithm may require significant computation, because it applies Bayes theorem to each hypothesis in H to calculate P(h/D).
- Impractical for large hypothesis spaces

Assumptions:

1. The training data D is noise free.
2. The target concept c is contained in the hypothesis space H
3. We have no a priori reason to believe that any hypothesis is more probable than any other.

Brute-Force Bayes Concept Learning

What values should we specify for P(h)?

- Given no prior knowledge that one hypothesis is more likely than another, it is reasonable to assign the same prior probability to every hypothesis h in H.

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

What choice shall we make for P(Dlh)?

- Since we assume noise-free training data, the probability of observing classification d_i given h is just 1 if d_i = h(x_i) and 0 if d_i != h(x_i). Therefore,

$$P(D|h) = \begin{cases} 1 \text{ if } d_i = h(x_i) \text{ for all } d_i \text{ in } D \\ 0 \text{ otherwise} \end{cases}$$

Brute-Force Bayes Concept Learning

- First consider the case where h is inconsistent with the training data D.

We have

$$P(h|D) = \frac{0 \cdot P(h)}{P(D)} = 0 \text{ if } h \text{ is inconsistent with } D$$

- Consider the case where h is consistent with D.

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

Brute-Force Bayes Concept Learning

We can derive P(D) from the theorem of total probability.

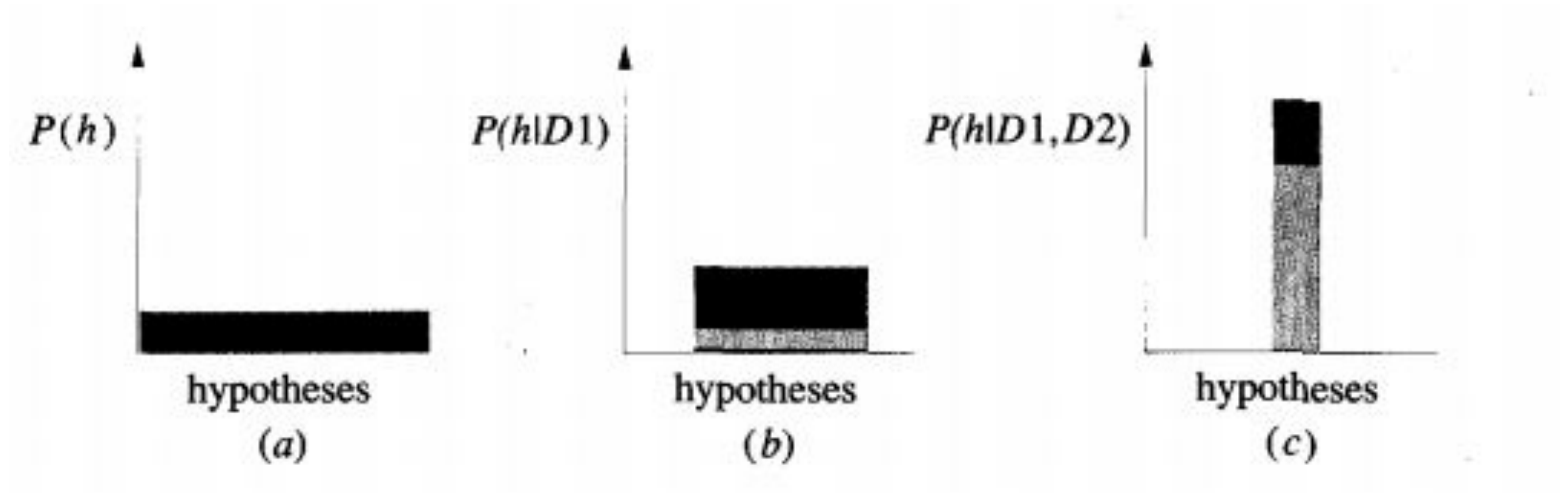
$$
\begin{aligned}
P(D) &= \sum_{h_i \in H} P(D|h_i)P(h_i) \\
&= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} + \sum_{h_i \notin VS_{H,D}} 0 \cdot \frac{1}{|H|} \\
&= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} \\
&= \frac{|VS_{H,D}|}{|H|}
\end{aligned}
$$

To summarize, Bayes theorem implies that the posterior probability P(h/D) under our assumed P(h) and P(D/h) is

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

Brute-Force Bayes Concept Learning

The evolution of probabilities associated with hypotheses is depicted schematically in the following Figure.



MAP Hypotheses and Consistent Learners

- In the given setting, every hypothesis consistent with D is a MAP hypothesis.
- A learning algorithm is a **consistent learner** provided it outputs a hypothesis that commits zero errors over the training examples.
- Every consistent learner outputs a MAP hypothesis, if we assume a uniform prior probability distribution over H and if we assume deterministic, noise free training data.
- FIND-S searches the hypothesis space H from specific to general hypotheses, outputting a maximally specific consistent hypothesis.
- Because FIND-S outputs a consistent hypothesis, it will output a MAP hypothesis under the probability distributions P(h) and P(D/h) defined above. Of course FIND-S does not explicitly manipulate probabilities at all-it simply outputs a maximally specific member of the version space.

MAP Hypotheses and Consistent Learners

- Because FIND-S outputs a maximally specific hypothesis from the version space, its output hypothesis will be a MAP hypothesis relative to any prior probability distribution that favors more specific hypotheses.
- More precisely, suppose $\mathcal{H}$ is any probability distribution P(h) over H that assigns $P(h_1) \geq P(h_2)$ if h_1 is more specific than h_2. Then it can be shown that FIND-S outputs a MAP hypothesis assuming the prior distribution $\mathcal{H}$ and the same distribution P(D/h).
- The Bayesian framework allows one way to characterize the behavior of learning algorithms even when the learning algorithm does not explicitly manipulate probabilities.

MAP Hypotheses and Consistent Learners

- Bayesian interpretation provides an alternative way to characterize the assumptions implicit in learning algorithms.
- Instead of modeling the inductive inference method by an equivalent deductive system, we model it by an equivalent probabilistic reasoning system based on Bayes theorem.
- And here the implicit assumptions that we attribute to the learner are assumptions of the form "the prior probabilities over H are given by the distribution P(h), and the strength of data in rejecting or accepting a hypothesis is given by P(D/h)."

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

- Consider the problem of learning a continuous-valued target function.
- A straightforward Bayesian analysis will show that under certain assumptions any learning algorithm that minimizes the squared error between the output hypothesis predictions and the training data will output a maximum likelihood hypothesis.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Consider the following problem setting:

Learner L considers an instance space X and a hypothesis space H consisting of some class of real-valued functions defined over X.

The problem faced by L is to learn an unknown target function $f : X \rightarrow \Re$

drawn from H.

A set of m training examples is provided, where the target value of each example is corrupted by random noise drawn according to a Normal probability distribution.

More precisely, each training example is a pair of the form (xi, di) where

$d_i = f(x_i) + e_i$.

It is assumed that the values of the ei are drawn independently and that they are distributed according to a Normal distribution with zero mean.

The task of the learner is to output a maximum likelihood hypothesis, or, equivalently, a MAP hypothesis assuming all hypotheses are equally probable a priori.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

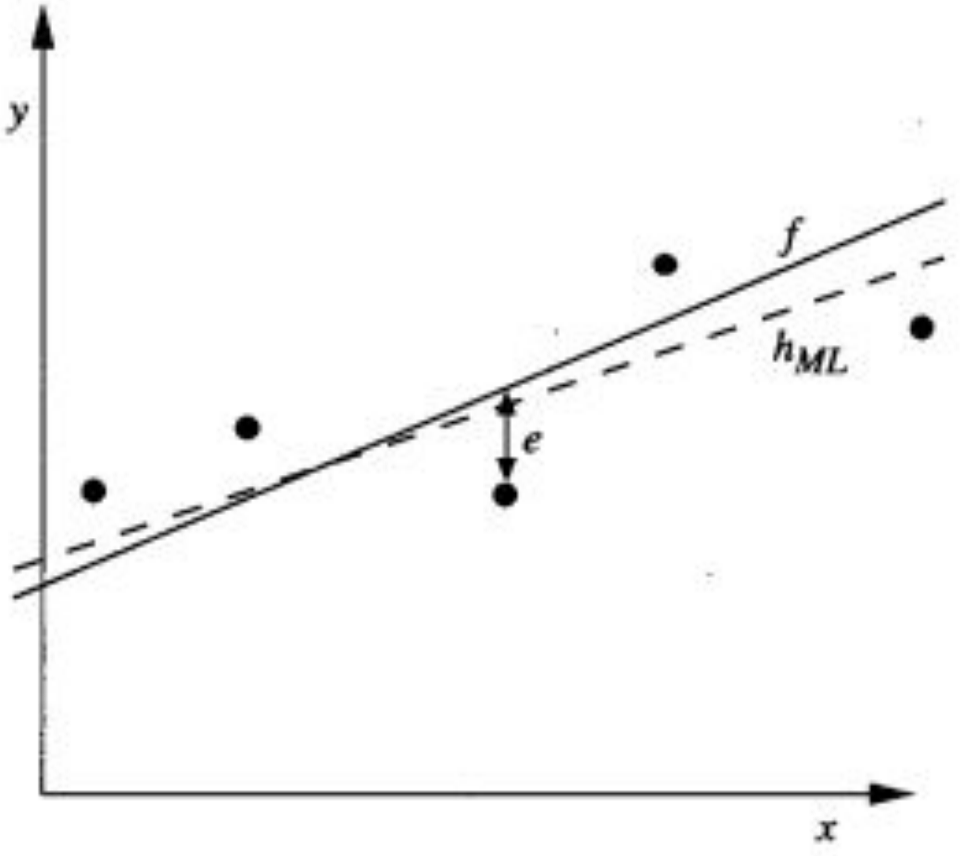


FIGURE 6.2
Learning a real-valued function. The target function f corresponds to the solid line. The training examples $\langle x_i, d_i \rangle$ are assumed to have Normally distributed noise e_i with zero mean added to the true target value $f(x_i)$. The dashed line corresponds to the linear function that minimizes the sum of squared errors. Therefore, it is the maximum likelihood hypothesis h_{ML}, given these five training examples.

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

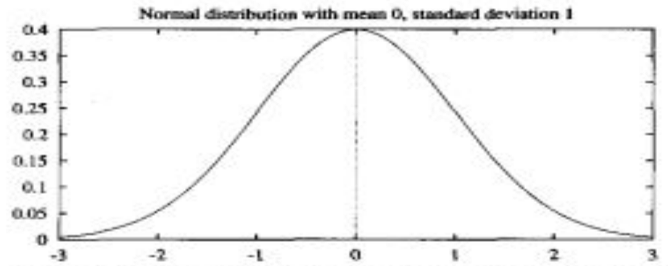
- First, in order to discuss probabilities over continuous variables such as e, we must introduce **probability densities**.
- The total probability over all possible values of the random variable to sum to one.
- In the case of continuous variables we cannot achieve this by assigning a finite probability to each of the infinite set of possible values for the random variable.
- Instead, we speak of a probability density for continuous variables such as e and require that the integral of this probability density over all possible values be one.

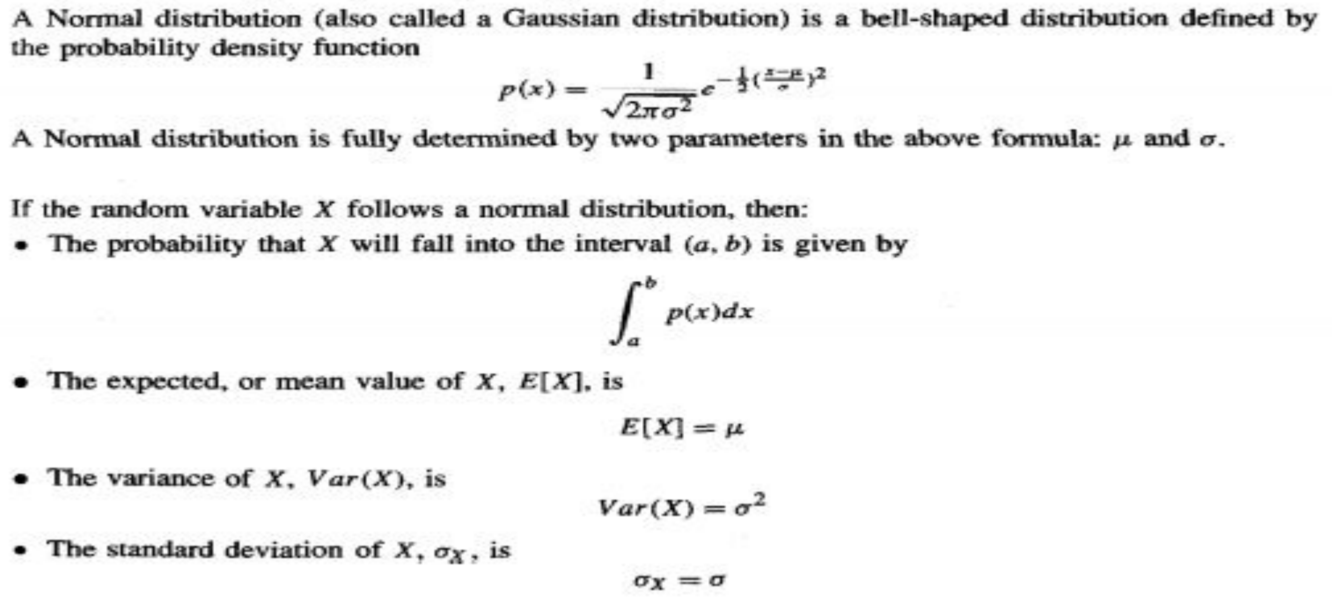
Probability density function:

$$p(x_0) \equiv \lim_{\epsilon \to 0} \frac{1}{\epsilon} P(x_0 \le x < x_0 + \epsilon)$$

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

- Second, the random noise variable e is generated by a Normal probability distribution.





A Normal distribution (also called a Gaussian distribution) is a bell-shaped distribution defined by the probability density function

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

A Normal distribution is fully determined by two parameters in the above formula: μ and σ.

If the random variable X follows a normal distribution, then:

- The probability that X will fall into the interval (a, b) is given by

$$\int_a^b p(x)dx$$

- The expected, or mean value of X, $E[X]$, is

$$E[X] = \mu$$

- The variance of X, $Var(X)$, is

$$Var(X) = \sigma^2$$

- The standard deviation of X, σ_X, is

$$\sigma_X = \sigma$$

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

- The maximum likelihood hypothesis is

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

- Assuming the training examples are mutually independent given h, we can write P(D/h) as the product of the various $p(d_i|h)$

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} p(d_i|h)$$

- Given that the noise ei obeys a Normal distribution

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(d_i - \mu)^2}$$

$$= \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(d_i - h(x_i))^2}$$

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

- Rather than maximizing the above complicated expression we shall choose to maximize its logarithm.

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

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} -\frac{1}{2\sigma^2}(d_i - h(x_i))^2$$

$$h_{ML} = \underset{h \in H}{\operatorname{argmin}} \sum_{i=1}^{m} \frac{1}{2\sigma^2}(d_i - h(x_i))^2$$

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

MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

Why is it reasonable to choose the Normal distribution to characterize noise?

- It allows for a mathematically straightforward analysis.
- The smooth, bell-shaped distribution is a good approximation to many types of noise in physical systems.

Limitations:

The above analysis considers noise only in the target value of the training example and does not consider noise in the attributes describing the instances themselves.

MAXIMUM LIKELIHOOD HYPOTHESES FOR PREDICTING PROBABILITIES

- Consider the setting in which we wish to learn a nondeterministic (probabilistic) function $f : X \rightarrow \{0, 1\}$, which has two discrete output values.
- Given this problem setting, we might wish to learn a neural network whose output is the probability that f (x) = 1.
- To learn the target function, $f' : X \rightarrow [0, 1]$, such that f '(x) = P(f (x) = 1)
- We can train a neural network directly from the observed training examples of f and derive a maximum likelihood hypothesis for f ‘.
- We must first obtain an expression for P(D/h).

MAXIMUM LIKELIHOOD HYPOTHESES FOR PREDICTING PROBABILITIES

- Assume the training data D is of the form D = $\{(x_1, d_1) \ldots (x_m, d_m)\}$, where d_i is the observed 0 or 1 value for f (x_i).
- We can write P(D/h) as

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

- The probability of encountering any particular instance x_i is independent of the hypothesis h.

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

MAXIMUM LIKELIHOOD HYPOTHESES FOR PREDICTING PROBABILITIES

- h is our hypothesis regarding the target function, which computes this very probability. Theref $P(d_i = 1|h, x_i) = h(x_i),$ and in general

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

- Re-express it in a more mathematically manipulable form

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

- Substitute it in the above equation,

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

MAXIMUM LIKELIHOOD HYPOTHESES FOR PREDICTING PROBABILITIES

- Now write an expression for the maximum likelihood hypothesis

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

- The last term is a constant independent of h, so it can be dropped

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

- It easier to work with the log of the likelihood, yielding

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

Gradient Search to Maximize Likelihood in a Neural Net

- We derive a weight-training rule for neural network learning that seeks to maximize G(h, D) using gradient ascent.

$$\begin{aligned}\frac{\partial G(h, D)}{\partial w_{jk}} &= \sum_{i=1}^{m} \frac{\partial G(h, D)}{\partial h(x_i)} \frac{\partial h(x_i)}{\partial w_{jk}} \\ &= \sum_{i=1}^{m} \frac{\partial(d_i \ln h(x_i) + (1 - d_i) \ln(1 - h(x_i)))}{\partial h(x_i)} \frac{\partial h(x_i)}{\partial w_{jk}} \\ &= \sum_{i=1}^{m} \frac{d_i - h(x_i)}{h(x_i)(1 - h(x_i))} \frac{\partial h(x_i)}{\partial w_{jk}}\end{aligned}$$

- Suppose our neural network is constructed from a single layer of sigmoid units.

$$\frac{\partial h(x_i)}{\partial w_{jk}} = \sigma'(x_i)x_{ijk} = h(x_i)(1 - h(x_i))x_{ijk}$$

Gradient Search to Maximize Likelihood in a Neural Net

- We obtain a simple expression for the derivatives that constitute the gradient

$$\frac{\partial G(h, D)}{\partial w_{jk}} = \sum_{i=1}^{m}(d_i - h(x_i))\ x_{ijk}$$

- Weight update rule

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

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

MINIMUM DESCRIPTION LENGTH PRINCIPLE

- The Minimum Description Length principle is motivated by interpreting the definition of h_{MAP} in the light of basic concepts from information theory.
- Consider definition of h_{MAP}

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

which can be equivalently expressed in terms of maximizing the $\log_2$

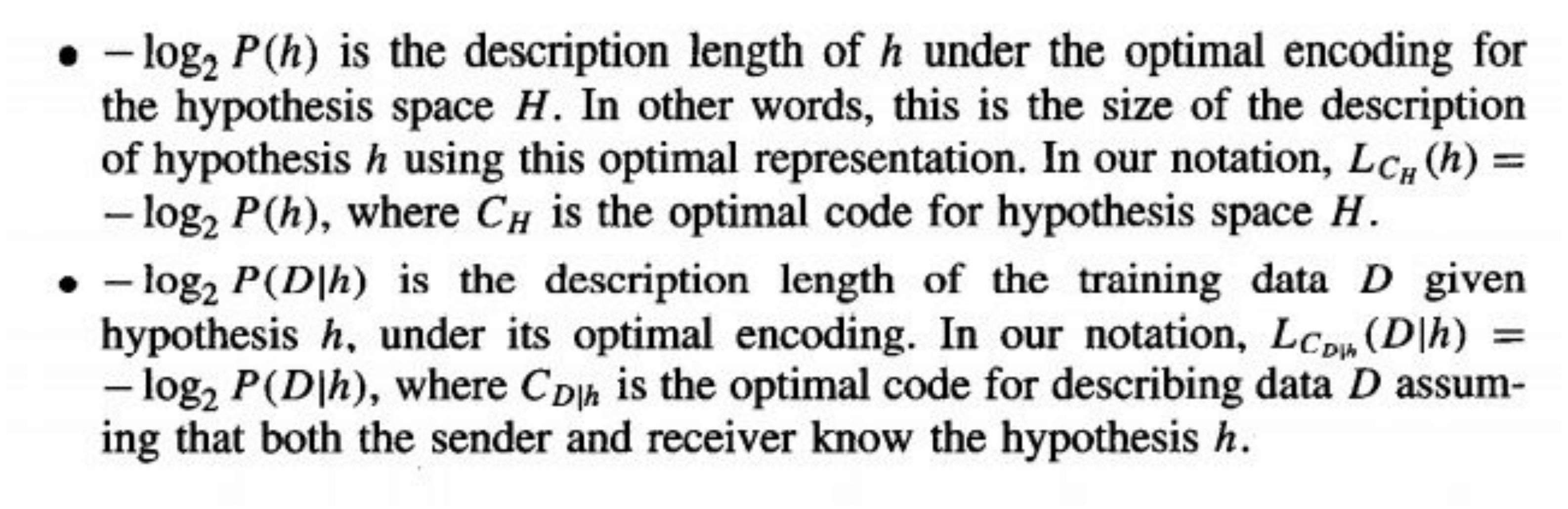
$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}}\ \log_2 P(D|h) + \log_2 P(h)$$

or alternatively, minimizing the negative of this quantity

$$h_{MAP} = \underset{h \in H}{\operatorname{argmin}}\ -\log_2 P(D|h) - \log_2 P(h)$$

MINIMUM DESCRIPTION LENGTH PRINCIPLE

- Interpret this Equation in the point of view of coding theory.



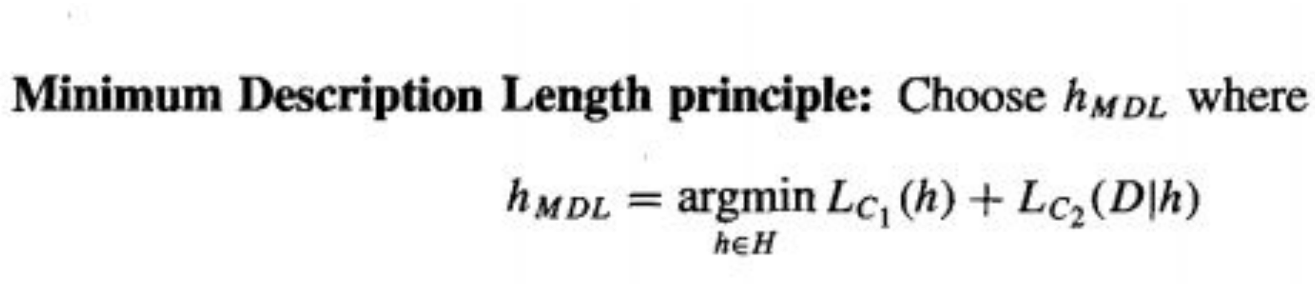
- $-\log_2 P(h)$ is the description length of h under the optimal encoding for the hypothesis space H. In other words, this is the size of the description of hypothesis h using this optimal representation. In our notation, $L_{C_H}(h) = -\log_2 P(h)$, where C_H is the optimal code for hypothesis space H.
- $-\log_2 P(D|h)$ is the description length of the training data D given hypothesis h, under its optimal encoding. In our notation, $L_{C_{D|h}}(D|h) = -\log_2 P(D|h)$, where $C_{D|h}$ is the optimal code for describing data D assuming that both the sender and receiver know the hypothesis h.

- We can rewrite the above Equation as

$$h_{MAP} = \underset{h}{\operatorname{argmin}}\, L_{C_H}(h) + L_{C_{D|h}}(D|h)$$

MINIMUM DESCRIPTION LENGTH PRINCIPLE

- Assuming we use the codes C_1 and C_2 to represent the hypothesis and the data given the hypothesis, we can state the MDL principle as



Minimum Description Length principle: Choose h_{MDL} where

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

Naïve Bayes Classifier

Naive Bayes Classifiers

Task: Classify a new instance based on a tuple of attribute values

$$\langle x_1, x_2, \ldots, x_n \rangle$$

$$c_{MAP} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j \mid x_1, x_2, \ldots, x_n)$$

$$c_{MAP} = \underset{c_j \in C}{\operatorname{argmax}} \frac{P(x_1, x_2, \ldots, x_n \mid c_j) P(c_j)}{P(x_1, x_2, \ldots, x_n)}$$

$$c_{MAP} = \underset{c_j \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c_j) P(c_j)$$

Naïve Bayes Classifier: Assumptions

- $P(c_j)$
  - Can be estimated from the frequency of classes in the training examples.
- $P(x_1,x_2,\ldots,x_n|c_j)$
  - $O(|X|^n \bullet |C|)$
  - Could only be estimated if a very, very large number of training examples was available.

Conditional Independence Assumption:

$\Rightarrow$ Assume that the probability of observing the conjunction of attributes is equal to the product of the individual probabilities.

Conditional Independence

Definition: X is <u>conditionally independent</u> of Y given Z, if the probability distribution governing X is independent of the value of Y, given the value of Z

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

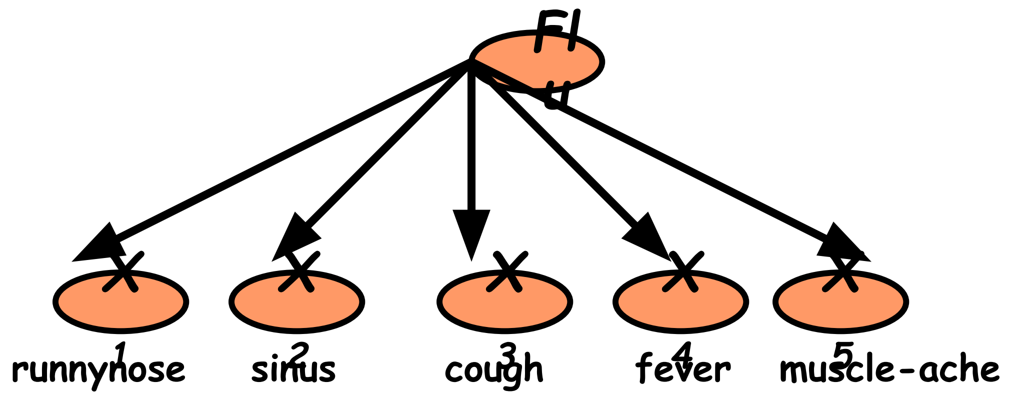
Which we often write

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

E.g.,

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

The Naïve Bayes Classifier



- **Conditional Independence Assumption:** features are independent of each other given the class:

$$P(X_1,\ldots,X_5 \mid C) = P(X_1 \mid C) \cdot P(X_2 \mid C) \cdot \cdots \cdot P(X_5 \mid C)$$

Naive Bayes Classifier

- It is easy to estimate each of the **$P(v_j)$** simply by counting the frequency with which each target value **v_j** occurs in the training data.
- However, estimating the different **$P(a_l, a_2 \ldots a_n \mid v_j)$** terms is not feasible unless we have a very, very large set of training data.
    - The problem is that the number of these terms is equal to the number of possible instances times the number of possible target values.
    - Therefore, we need to see every instance in the instance space many times in order to obtain reliable estimates.
- The naive Bayes classifier is based on the simplifying assumption that the attribute values are conditionally independent given the target value.
- For a given the target value of the instance, the probability of observing conjunction ***$a_l, a_2 \ldots a_n$***, is just the product of the probabilities for the individual attributes:

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

- **Naive Bayes classifier:**

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

Naive Bayes Classifier - Ex

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Naive Bayes Classifier -Ex

- New instance to classify:

  (Outlook=sunny, Temperature=cool, Humidity=high, Wind=strong)
- Our task is to predict the target value (yes or no) of the target concept *PlayTennis* for this new instance.

$$v_{NB} = \underset{v_j \in \{yes,no\}}{\operatorname{argmax}} P(v_j) \prod_i P(a_i|v_j)$$

$$= \underset{v_j \in \{yes,no\}}{\operatorname{argmax}} P(v_j) \; \mathbf{P(Outlook=sunny|v_j)\, P(Temperature=cool|v_j)\, P(Humidity=high|v_j)\, P(Wind=strong|v_j)}$$

Naive Bayes Classifier -Ex

- P(PlayTennis = yes) = 9/14 = .64
- P(PlayTennis = no) = 5/14 = .36
- P(sunny/yes)= 2/9
- P(sunny/no) = 3/5
- P(cool/yes) = 3/9
- P(cool/no) = 1/5
- P(high/yes)= 3/9
- P(high/no)= 4/5
- P(strong/yes)= 3/9
- P(strong/no)= 3/5

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Naive Bayes Classifier -Ex

- Likelihood of yes $= \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} = 0.0053$

- Likelihood of no $= \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} = 0.0206$

$$P(yes)\ P(sunny|yes)\ P(cool|yes)\ P(high|yes)\ P(strong|yes) = .0053$$

$$P(no)\ P(sunny|no)\ P(cool|no)\ P(high|no)\ P(strong|no) = .0206$$

Thus, the naive Bayes classifier assigns the target value ***PlayTennis*** = ***no*** to this new instance, based on the probability estimates learned from the training data.

- Furthermore, by normalizing the above quantities to sum to one we can calculate the conditional probability that the target value is ***no,*** given the observed attribute values.

.0206 / (.0206 + .0053) = .795

Estimating Probabilities

- **P(Wind**=**strong | PlayTennis**=**no)** by the fraction **n_c/n** where ***n*** = 5 is the total number of training examples for which **PlayTennis**=**no,** and ***n_c*** = 3 is the number of these for which **Wind**=**strong.**
- When ***n_c*** is zero
  - **n_c/n** will be zero too
  - this probability term will dominate
- To avoid this difficulty we can adopt a Bayesian approach to estimating the probability, using the m-estimate defined as follows.

  m-estimate of probability: (n_c+ m*p) / (n + m)

- if an attribute has ***k*** possible values we set p = 1/k .
  - p=0.5 because Wind has two possible values.
- m is called the equivalent sample size
  - augmenting the ***n*** actual observations by an additional m virtual samples distributed according to p.

Bayesian Belief Networks

Bayesian Belief Networks

- Naïve Bayes is based on assumption of conditional independence.
- This assumption reduces the complexity of learning the target function.
- When it is met, the naive Bayes classifier outputs the optimal Bayes classification.
- Bayesian belief networks provide a tractable method for specifying dependencies among variables.
- A Bayesian belief network describes the probability distribution governing a set of variables by specifying a set of conditional independence assumptions along with a set of conditional probabilities.

Bayesian Belief Networks

- Bayesian belief networks allow stating conditional independence assumptions that apply to subsets of the variables.
- Bayesian belief networks provide an intermediate approach that is less constraining than the global assumption of conditional independence made by the naive Bayes classifier, but more tractable than avoiding conditional independence assumptions altogether.

Terminology

- A Bayesian Belief Network describes the probability distribution over a set of random variables $Y_1, Y_2, \ldots Y_n$ where Each variable Y_i can take on the set of values $V(Y_i)$
- **The joint space** of the set of variables Y is the cross product

  $V(Y_1) \times V(Y_2) \times \ldots \times V(Y_n)$
- Each item in the joint space corresponds to one possible assignment of values to the tuple of variables $<Y_1, \ldots Y_n>$
- **Joint probability distribution**: specifies the probabilities of the items in the joint space
- **A Bayesian Network** provides a way to describe the joint probability distribution in a compact manner.

Conditional Independence

- Let X, Y, and Z be three discrete-valued random variables.
- We say that X is conditionally independent of Y given Z if the probability distribution governing X is independent of the value of Y given a value for Z

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

$$P(X \mid Y, Z) = P(X \mid Z)$$

Conditional Independence

- This definition of conditional independence can be extended to sets of variables.
- We say that the set of variables $X_1, X_2, \ldots X_l$ is conditionally independent of the set of variables $Y_1, Y_2, \ldots Y_m$ given the set of variables $Z_1, Z_2, \ldots Z_n$, if

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

Bayesian Belief Network Representation

A Bayesian belief network (Bayesian network for short) represents the joint probability distribution for a set of variables

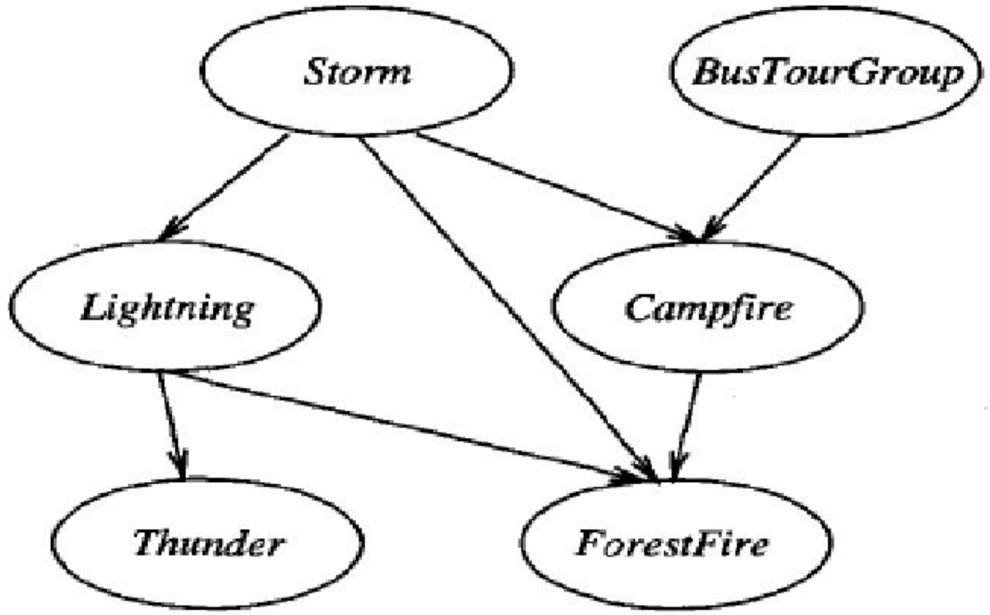
A set of random variables makes up the **nodes** of the network.

For each variable two types of information are specified.

- First, the network arcs represent the assertion that the variable is conditionally independent of its non-descendants in the network given its immediate predecessors in the network.
- A set of directed links or arrows connects pairs of nodes. The intuitive meaning of an arrow from X to Y is that X has a direct influence on Y.
- Each node has a conditional probability table that quantifies the effects that the parents have on the node. The parents of a node are all those nodes that have arrows pointing to it.

The graph has no directed cycles (it is a DAG).

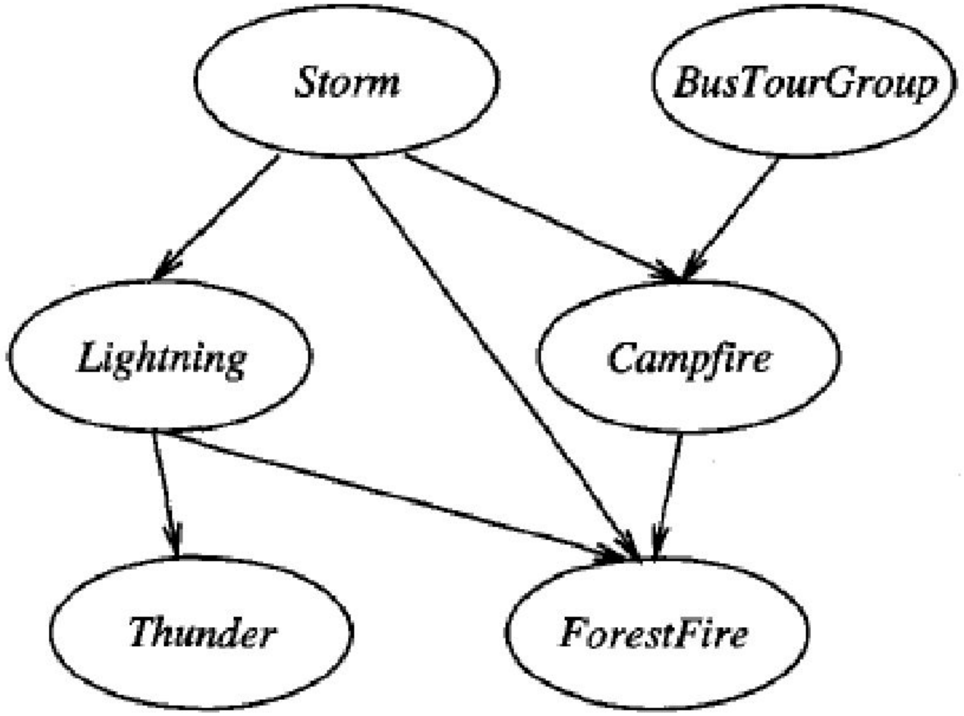
Bayesian Belief Network Example



	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

FIGURE 6.3
A Bayesian belief network. The network on the left represents a set of conditional independence assumptions. In particular, each node is asserted to be conditionally independent of its nondescendants, given its immediate parents. Associated with each node is a conditional probability table, which specifies the conditional distribution for the variable given its immediate parents in the graph. The conditional probability table for the *Campfire* node is shown at the right, where *Campfire* is abbreviated to *C*, *Storm* abbreviated to *S*, and *BusTourGroup* abbreviated to *B*.

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

$$P(Campfire = True | Storm = True, BusTourGroup = True) = 0.4$$

Inference

- A Bayesian network is used to infer the value of some target variable (e.g., ForestFire) given the observed values of the other variables.
- A Bayesian network is used to infer is the probability distribution for the target variable, which specifies the probability that it will take on each of its possible values given the observed values of the other variables.
- This inference step can be straightforward if values for all of the other variables in the network are known exactly.
- In the more general case we may wish to infer the probability distribution for some variable (e.g., ForestFire) given observed values for only a subset of the other variables (e.g., Thunder and BusTourGroup may be the only observed values available).
- In general, a Bayesian network can be used to compute the probability distribution for any subset of network variables given the values or distributions for any subset of the remaining variables.

Learning Bayesian Belief Networks

Several different settings for this learning problem can be considered.

- First, the network structure might be given in advance, or it might have to be inferred from the training data.
- Second, all the network variables might be directly observable in each training example, or some might be unobservable.

In the case where the network structure is given in advance and the variables are fully observable in the training examples, learning the conditional probability tables is straightforward.

In the case where the network structure is given but only some of the variable values are observable in the training data, the learning problem is more difficult.

The gradient ascent procedure searches through a space of hypotheses that corresponds to the set of all possible entries for the conditional probability tables.

Gradient Ascent Training of Bayesian Networks

- The gradient of ln P(D/h) is given by the derivatives $\frac{\partial \ln P(D|h)}{\partial w_{ijk}}$ for each of the w_{ijk}.
- Each of these derivatives can be calculated as

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

- To simplify notation, in this derivation we will write the abbreviation $P_h(D)$ to represent P(D/h).

Gradient Ascent Training of Bayesian Networks

- Assuming the training examples d in the data set D are drawn independently, we write this derivative as

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

since $\frac{\partial \ln f(x)}{\partial x} = \frac{1}{f(x)} \frac{\partial f(x)}{\partial x}$

Gradient Ascent Training of Bayesian Networks

By summing over their possible values $y_{ij'}$ and $u_{ik'}$,

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

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

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

$$\frac{\partial}{\partial w_{ijk}}$$

Given that the only term in this sum for which

is nonzero is the e

$$\frac{\partial \ln P_h(D)}{\partial w_{ijk}} = \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} P_h(d|y_{ij}, u_{ik}) P_h(y_{ij}|u_{ik}) P_h(u_{ik})$$

$$= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} P_h(d|y_{ij}, u_{ik}) w_{ijk} P_h(u_{ik})$$

$$= \sum_{d \in D} \frac{1}{P_h(d)} P_h(d|y_{ij}, u_{ik}) P_h(u_{ik})$$

Gradient Ascent Training of Bayesian Networks

- Applying Bayes theorem to rewrite $P_h(d|y_{ij}, u_{ik})$, we have

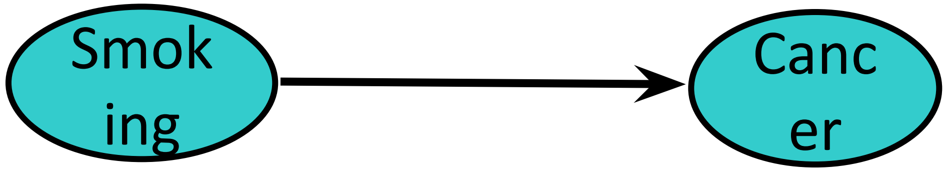
$$\begin{aligned}\frac{\partial \ln P_h(D)}{\partial w_{ijk}} &= \sum_{d\in D} \frac{1}{P_h(d)} \frac{P_h(y_{ij}, u_{ik}|d) P_h(d) P_h(u_{ik})}{P_h(y_{ij}, u_{ik})} \\ &= \sum_{d\in D} \frac{P_h(y_{ij}, u_{ik}|d) P_h(u_{ik})}{P_h(y_{ij}, u_{ik})} \\ &= \sum_{d\in D} \frac{P_h(y_{ij}, u_{ik}|d)}{P_h(y_{ij}|u_{ik})}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

- Weight updating can be done using the following rule:

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

Bayesian Networks

$S \in \{no, light, heavy\}$

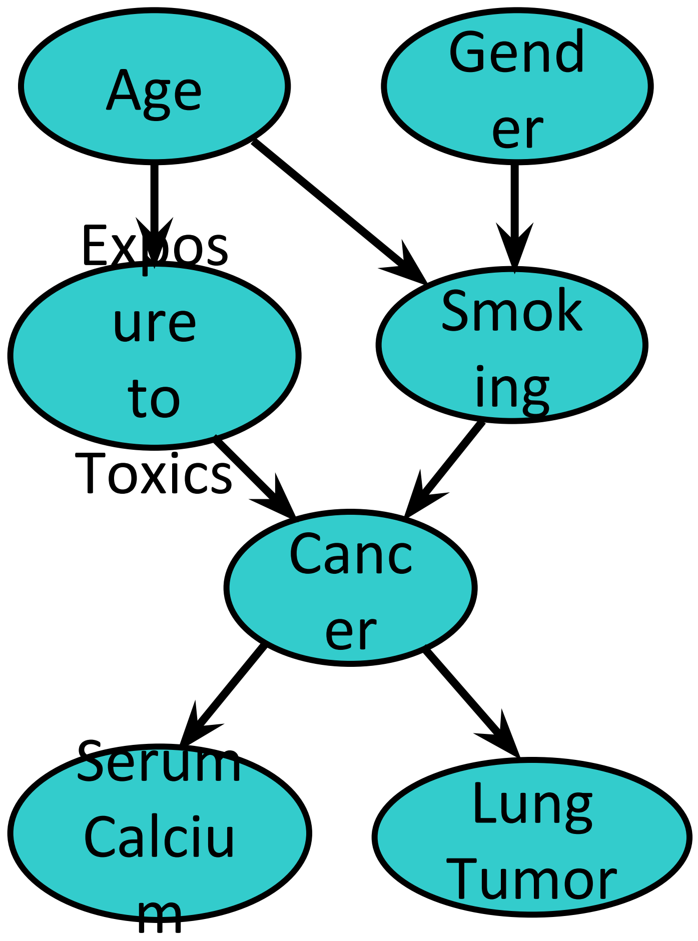


$C \in \{none, benign, malignant\}$

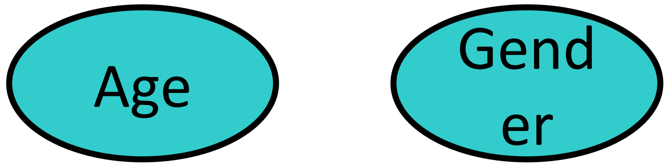
P(S=no)	0.80
P(S=light)	0.15
P(S=heavy)	0.05

Smoking=	no	light	heavy
P(C=none)	0.96	0.88	0.60
P(C=benign)	0.03	0.08	0.25
P(C=malig)	0.01	0.04	0.15

A Bayesian Network



Independence



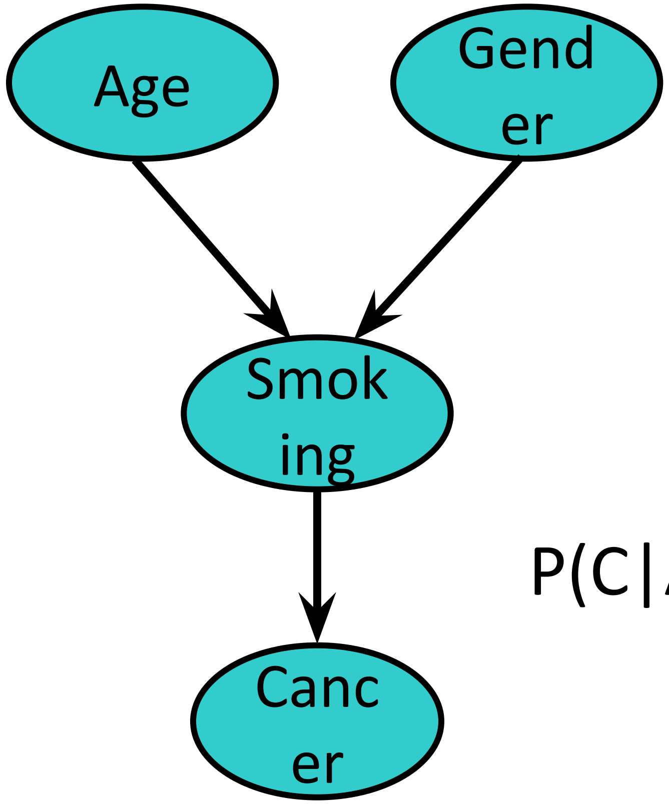
Age and Gender are independent.

$$P(A,G) = P(G)P(A)$$

$$P(A|G) = P(A) \quad A \perp G$$
$$P(G|A) = P(G) \quad G \perp A$$

$$P(A,G) = P(G|A)\,P(A) = P(G)P(A)$$
$$P(A,G) = P(A|G)\,P(G) = P(A)P(G)$$

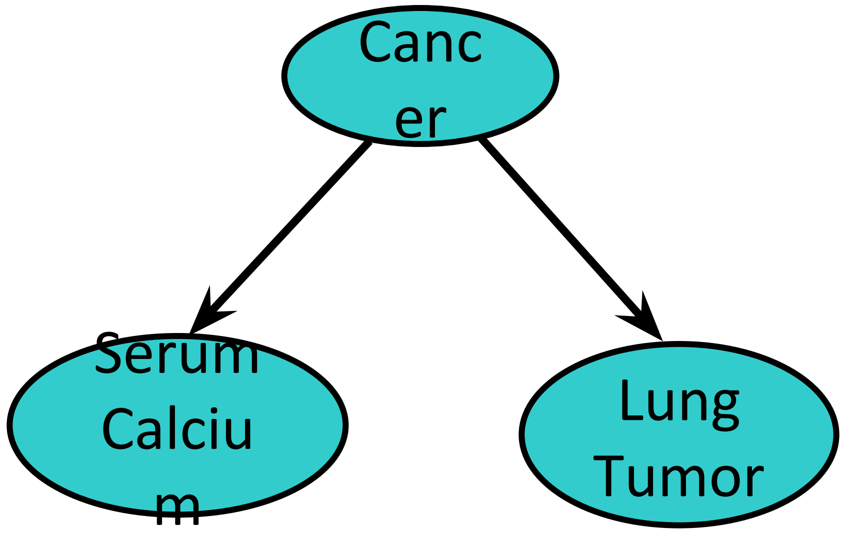
Conditional Independence



Cancer is independent of Age and Gender given Smoking.

$P(C|A,G,S) = P(C|S)$ $C \perp A,G \mid S$

More Conditional Independence: Naïve Bayes

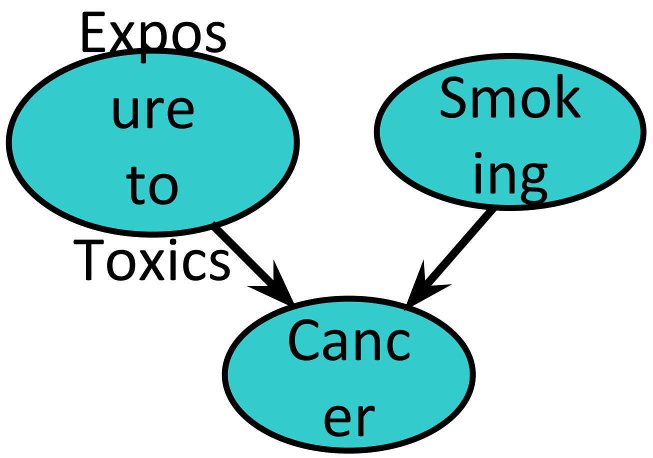


Serum Calcium and Lung Tumor are dependent

Serum Calcium is independent of Lung Tumor, given Cancer

$$P(L|SC,C) = P(L|C)$$

More Conditional Independence: Explaining Away



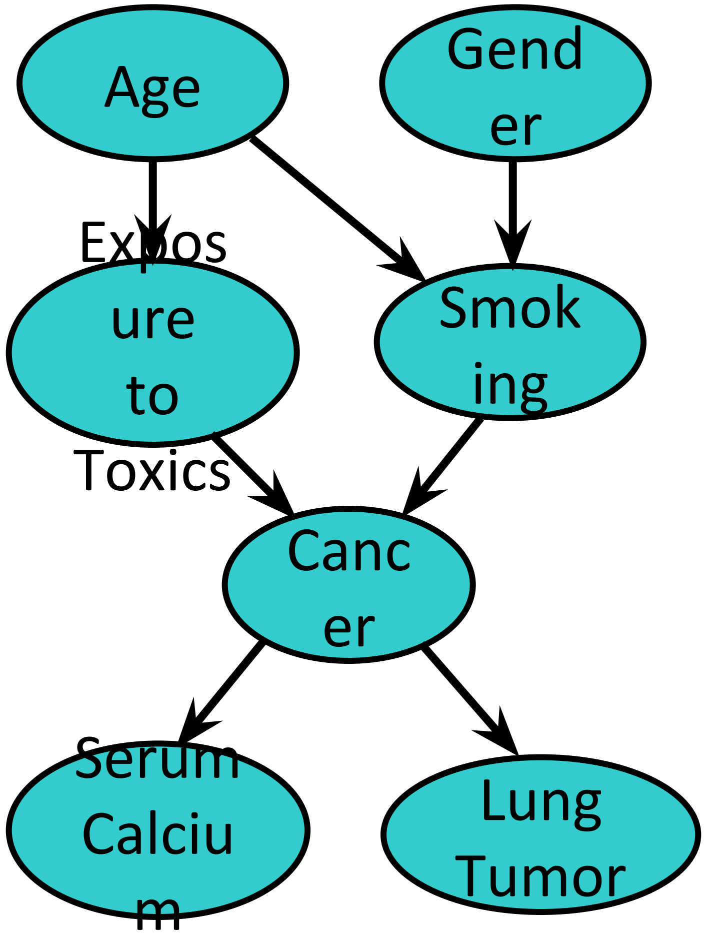
Exposure to Toxics and Smoking are independent

$E \perp S$

Exposure to Toxics is **dependent** on Smoking, given Cancer

$$P(E = \text{heavy} \mid C = \text{malignant}) > P(E = \text{heavy} \mid C = \text{malignant}, S = \text{heavy})$$

Put it all together



$$P(A,G,E,S,C,L,SC) =$$

$$P(A)\cdot P(G)\cdot$$

$$P(E \mid A)\cdot P(S \mid A,G)\cdot$$

$$P(C \mid E,S)\cdot$$

$$P(SC \mid C)\cdot P(L \mid C)$$

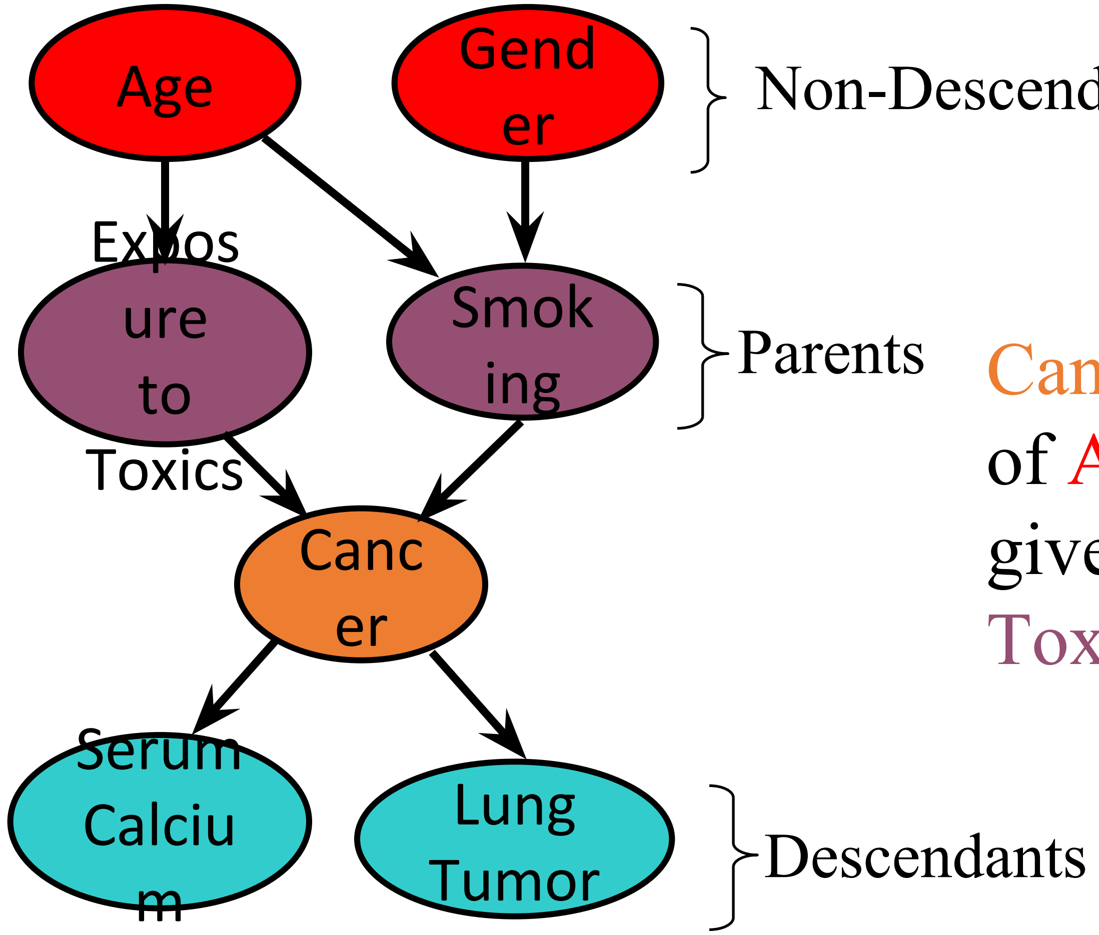
General Product (Chain) Rule for Bayesian Networks

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \boldsymbol{Pa_i})$$

Pa$_{\mathbf{i}}$=parents(X$_i$)

Conditional Independence

A variable (node) is conditionally independent of its non-descendants given its parents.



Cancer is independent of Age and Gender given Exposure to Toxics and Smoking.

THE EM ALGORITHM

EM Algorithm is used in the following situations:

- To learn in the presence of unobserved variables.
- For variables whose value is never directly observed, provided the general form of the probability distribution governing these variables is known.
- To train Bayesian belief networks as well as radial basis function networks.
- Provides basis for many unsupervised clustering algorithms and Baum-Welch forward-backward algorithm for learning Partially Observable Markov Models.

THE EM ALGORITHM

Estimating Means of k Gaussians:

Consider a problem in which the data D is a set of instances generated by a probability distribution that is a mixture of k distinct Normal distributions.

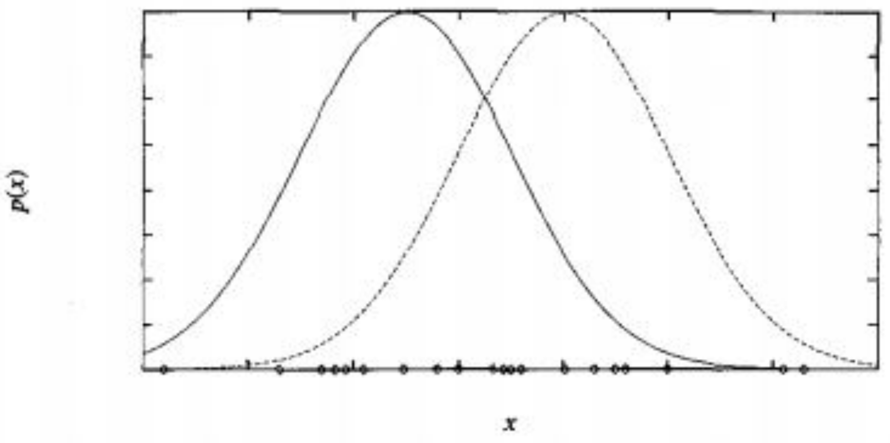


FIGURE 6.4
Instances generated by a mixture of two Normal distributions with identical variance σ. The instances are shown by the points along the x axis. If the means of the Normal distributions are unknown, the EM algorithm can be used to search for their maximum likelihood estimates.

THE EM ALGORITHM

Estimating Means of k Gaussians:

Each instance is generated using a two-step process.

- One of the k Normal distributions is selected at random.
- A single random instance x_i is generated according to this selected distribution.

This process is repeated to generate a set of data points.

Consider the special case where the selection of the single Normal distribution at each step is based on choosing each with uniform probability, where each of the k Normal distributions has the same variance σ^2 , and where σ^2 is known.

The learning task is to output a hypothesis h = $(\mu_1,\ldots\ldots,\mu_k)$ that describes the means of each of the k distributions. We would like to find a maximum likelihood hypothesis for these means; that is, a hypothesis h that maximizes p(D/h).

THE EM ALGORITHM

Estimating Means of k Gaussians:

The maximum likelihood hypothesis is the one that minimizes the sum of squared errors over the m training instances.

$$\mu_{ML} = \underset{\mu}{\operatorname{argmin}} \sum_{i=1}^{m} (x_i - \mu)^2$$

The sum of squared errors is minimized by the sample mean

$$\mu_{ML} = \frac{1}{m} \sum_{i=1}^{m} x_i$$

But this problem involves a mixture of k different Normal distributions, and we cannot observe which instances were generated by which distribution.

Thus, we have an example of a problem involving hidden variables.

THE EM ALGORITHM

Estimating Means of k Gaussians:

Consider each instance as the triple (x_i, z_{i1} , z_{i2}), where x_i is the observed value of the ith instance and where z_{i1} and z_{i2} indicate which of the two Normal distributions was used to generate the value xi.

In particular, z_{ij} has the value 1 if x_i was created by the jth Normal distribution and 0 otherwise.

Here x_i is the observed variable in the description of the instance, and z_{i1} and z_{i2} are hidden variables.
If the values of z_{i1} and z_{i2} were observed, the following Equation can be used to solve for the means p1 and p2. Because they are not, we will instead use the EM algorithm.

$$\mu_{ML} = \underset{\mu}{\operatorname{argmin}} \sum_{i=1}^{m} (x_i - \mu)^2$$

THE EM ALGORITHM

Estimating Means of k Gaussians:

The EM algorithm first initializes the hypothesis to $h = (\mu_1, \mu_2)$, where μ_1 and μ_2 are arbitrary initial values. It then iteratively re-estimates h by repeating the following two steps until the procedure converges to a stationary value for h.

Step 1: Calculate the expected value $E[z_{ij}]$ of each hidden variable z_{ij}, assuming the current hypothesis $h = (\mu_1, \mu_2)$ holds.

Step 2: Calculate a new maximum likelihood hypothesis $h' = (\mu_1', \mu_2')$, assuming the value taken on by each hidden variable z_{ij} is its expected value $E[z_{ij}]$ calculated in Step 1. Then replace the hypothesis $h = (\mu_1, \mu_2)$ by the new hypothesis $h' = (\mu_1', \mu_2')$ and iterate.

THE EM ALGORITHM

Estimating Means of k Gaussians:

Step 1 must calculate the expected value of each z_{ij}. This $E[z_{ij}]$ is just the probability that instance x_i was generated by the jth Normal distribution.

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

$$= \frac{e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)^2}}{\sum_{n=1}^{2} e^{-\frac{1}{2\sigma^2}(x_i - \mu_n)^2}}$$

In the second step we use the $E[z_{ij}]$ calculated during Step 1 to derive a new maximum likelihood hypothesis $h' = (\mu_1', \mu_2')$.

The maximum likelihood hypothesis in this case is given by

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

THE EM ALGORITHM

In its general form, the EM algorithm repeats the following two steps until convergence:

Step 1: Estimation (E) step: Calculate Q(h'/h) using the current hypothesis h and the observed data X to estimate the probability distribution over Y.

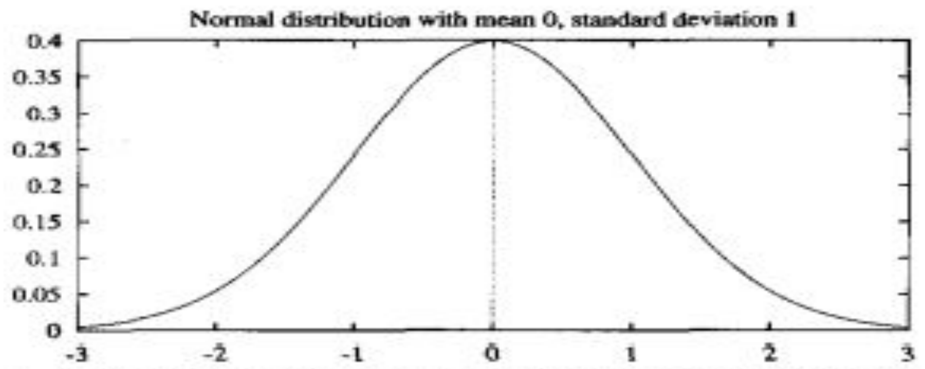
$$Q(h'|h) \leftarrow E[\ln P(Y|h')|h, X]$$

Step 2: Maximization (M) step: Replace hypothesis h by the hypothesis h' that maximizes this Q function.

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

When the function Q is continuous, the EM algorithm converges to a stationary point of the likelihood function P(Y/h'). When this likelihood function has a single maximum, EM will converge to this global maximum likelihood estimate for h'. Otherwise, it is guaranteed only to converge to a local maximum.

NORMAL DISTRIBUTION



A Normal distribution (also called a Gaussian distribution) is a bell-shaped distribution defined by the probability density function

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

A Normal distribution is fully determined by two parameters in the above formula: μ and σ.

If the random variable X follows a normal distribution, then:

- The probability that X will fall into the interval (a, b) is given by

$$\int_a^b p(x)dx$$

- The expected, or mean value of X, $E[X]$, is

$$E[X] = \mu$$

- The variance of X, $Var(X)$, is

$$Var(X) = \sigma^2$$

- The standard deviation of X, σ_X, is

$$\sigma_X = \sigma$$

Derivation of the k Means Algorithm

To apply EM we must derive an expression for Q(h/h’) that applies to our k-means problem.

First, derive an expression for ln p(Y/h’). Note the probability $p(y_i /h')$ of a single instance $y_i = (x_i , z_{i1}, \ldots z_{ik})$ of the full data can be written

$$p(y_i|h') = p(x_i, z_{i1}, \ldots, z_{ik}|h') = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}\sum_{j=1}^{k} z_{ij}(x_i-\mu_j')^2}$$

Given this probability for a single instance $p(y_i /h')$, the logarithm of the probability $\ln p(y_i /h')$ for all m instances in the data is

$$\begin{aligned}
\ln P(Y|h') &= \ln \prod_{i=1}^{m} p(y_i|h') \\
&= \sum_{i=1}^{m} \ln p(y_i|h') \\
&= \sum_{i=1}^{m} \left(\ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^{k} z_{ij}(x_i - \mu_j')^2 \right)
\end{aligned}$$

Derivation of the k Means Algorithm

The above expression for ln P(Y/h') is a linear function of these z_{ij}.

In general, for any function f (z) that is a linear function of z, the following equality holds

$$E[f(z)] = f(E[z])$$

This general fact about linear functions allows us to write

$$\begin{aligned} E[\ln P(Y|h')] &= E\left[\sum_{i=1}^{m}\left(\ln\frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2}\sum_{j=1}^{k} z_{ij}(x_i - \mu'_j)^2\right)\right] \\ &= \sum_{i=1}^{m}\left(\ln\frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2}\sum_{j=1}^{k} E[z_{ij}](x_i - \mu'_j)^2\right) \end{aligned}$$

To summarize, the function Q(h'/h) for the k means problem is

$$Q(h'|h) = \sum_{i=1}^{m}\left(\ln\frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2}\sum_{j=1}^{k} E[z_{ij}](x_i - \mu'_j)^2\right)$$

Derivation of the k Means Algorithm

$E[z_{ij}]$ is calculated based on the current hypothesis h and observed data X.

$$E[z_{ij}] = \frac{e^{-\frac{1}{2\sigma^2}(x_i-\mu_j)^2}}{\sum_{n=1}^{k} e^{-\frac{1}{2\sigma^2}(x_i-\mu_n)^2}}$$

Thus, the first (estimation) step of the EM algorithm defines the Q function based on the estimated $E[z_{ij}]$ terms. The second (maximization) step then finds the values μ_1',……..,μ_k' that maximize this Q function.

In the current case

$$\underset{h'}{\operatorname{argmax}}\, Q(h'|h) = \underset{h'}{\operatorname{argmax}} \sum_{i=1}^{m} \left(\ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^{k} E[z_{ij}](x_i - \mu_j')^2 \right)$$

$$= \underset{h'}{\operatorname{argmin}} \sum_{i=1}^{m} \sum_{j=1}^{k} E[z_{ij}](x_i - \mu_j')^2$$

Derivation of the k Means Algorithm

Thus, the maximum likelihood hypothesis here minimizes a weighted sum of squared errors, where the contribution of each instance x_i to the error that defines μ_j' is weighted by $E[z_{ij}]$.

The quantity given by the above Equation is minimized by setting each μ_j' to the weighted sample mean

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