BAYESIAN LEARNING

OVERVIEW

- Introduction
- Bayes theorem
- Bayes theorem and concept learning
- Maximum likelihood and Least Squared Error Hypothesis
- Maximum likelihood Hypotheses for predicting probabilities
- Minimum Description Length Principle
- Naive Bayes classifier
- Bayesian belief networks
- EM algorithm

INTRODUCTION

- Bayesian reasoning provides a probabilistic approach to inference.
- It is based on the assumption that the quantities of interest are governed by probability distributions and that optimal decisions can be made by reasoning about these probabilities together with observed data.
- It is important to machine learning because it provides a quantitative approach to weighing the evidence supporting alternative hypotheses.
- Bayesian reasoning provides the basis for learning algorithms that directly manipulate probabilities, as well as a framework for analyzing the operation of other algorithms that do not explicitly manipulate probabilities.

INTRODUCTION

- **Probability?**
  - **Probability** is the branch of mathematics concerning numerical descriptions of how likely an event is to occur/not occur.
  - The **probability** of an event is a number between 0 and 1, where, roughly speaking, 0 indicates impossibility of the event (not occur)and 1 indicates certainty(occur).
- **Inference?**
  - a conclusion reached on the basis of evidence and reasoning/ conclusion made based on observing the dataset/ **Inference** is using observation and background to reach a logical conclusion.
- **Hypothesis?**
  - A proposed explanation/statement made on the basis of limited evidence(data sets) as a starting point for further investigation.
  - Null Hypothesis(NO), Alternative Hypothesis(YES).
- **Prior Probability?**
  - General truth related to the fact.
  - **Prior probability** represents what is originally believed before new evidence is introduced

INTRODUCTION

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

- First, 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.

- The second reason is that they provide a useful perspective for understanding many learning algorithms that do not explicitly manipulate probabilities.

FEATURES OF BAYESIAN LEARNING METHODS

1. 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
2. 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, and

   (2) a probability distribution over observed data for each possible hypothesis.
3. Bayesian methods can accommodate hypotheses that make probabilistic predictions
4. New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities.
5. 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.

PRACTICAL DIFFICULTY IN APPLYING BAYESIAN METHODS

- One practical difficulty in applying Bayesian methods is that 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.

- A second practical difficulty is the significant computational cost required to determine the Bayes optimal hypothesis in the general case. In certain specialized situations, this computational cost can be significantly reduced.

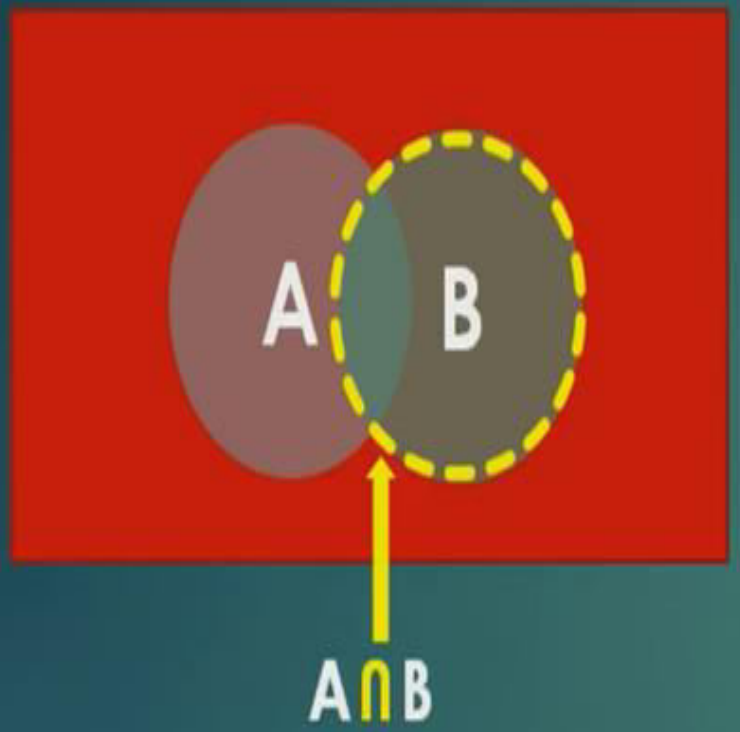
BAYES THEOREM

- **Bayes' Theorem** (also known as Bayes' rule) is a deceptively simple formula used to calculate **conditional probability**.
- The Theorem was named after English mathematician **Thomas Bayes** (1701-1761). The formal definition for the rule:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

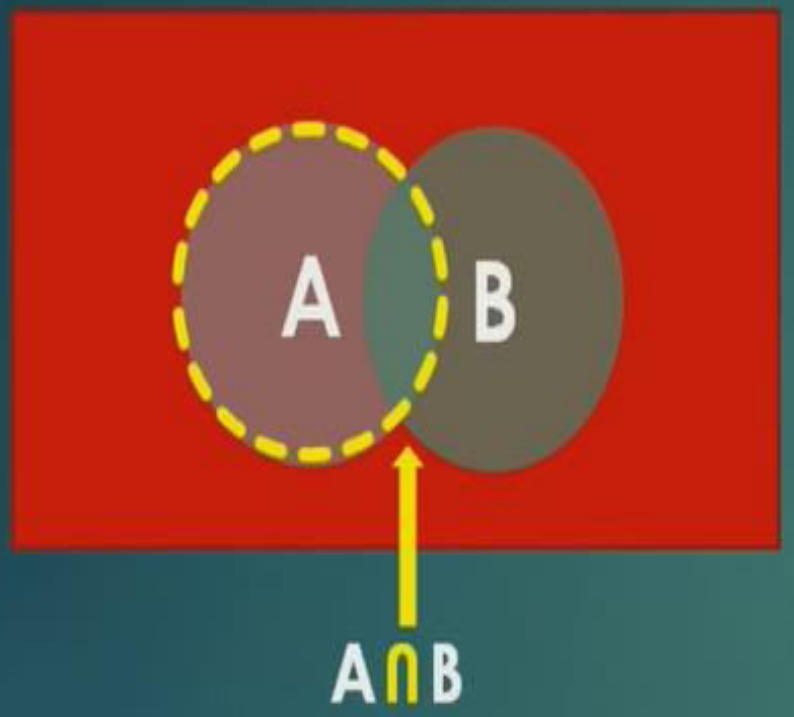
- **Bayes Theorem** is also widely used in the field of machine learning.
- Including its use in a probability framework for fitting a model to a training dataset, referred to as *maximum a posteriori or MAP* for short, and in developing models for *classification predictive modeling* problems such as the *Bayes Optimal Classifier* and *Naive Bayes*.

Conditional Probability



$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

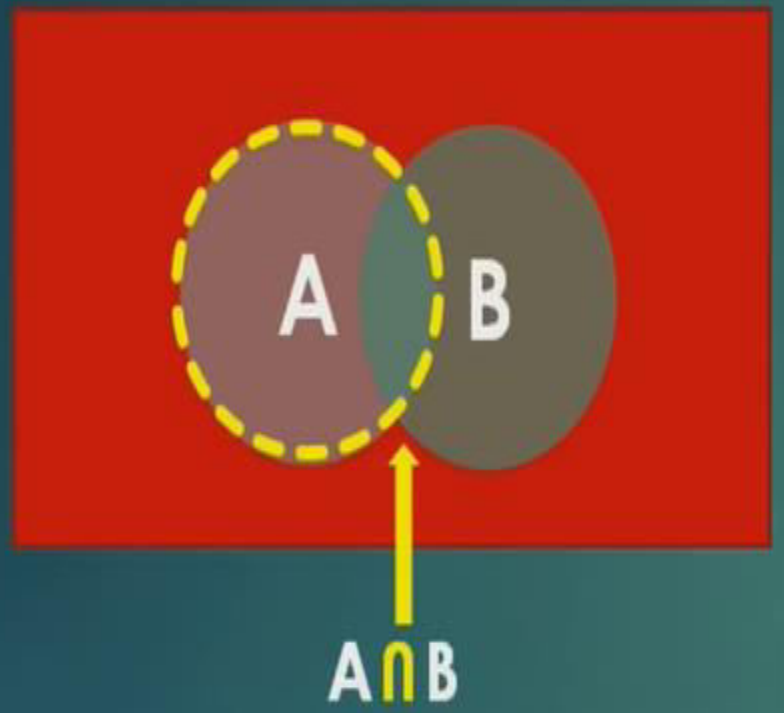
Conditional Probability



$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)}$$

Conditional Probability

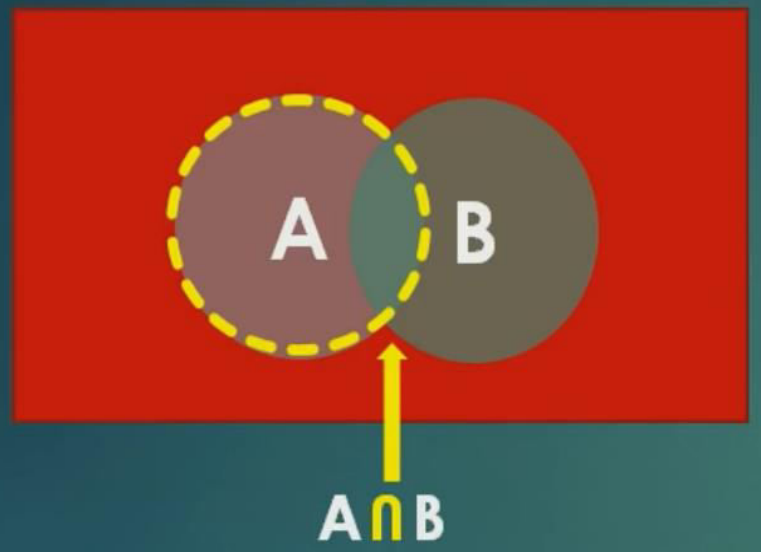


$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)}$$

$$P(A \cap B) = P(A \mid B) * P(B) = P(B \mid A) * P(A) \quad ---(i)$$

Conditional Probability



$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

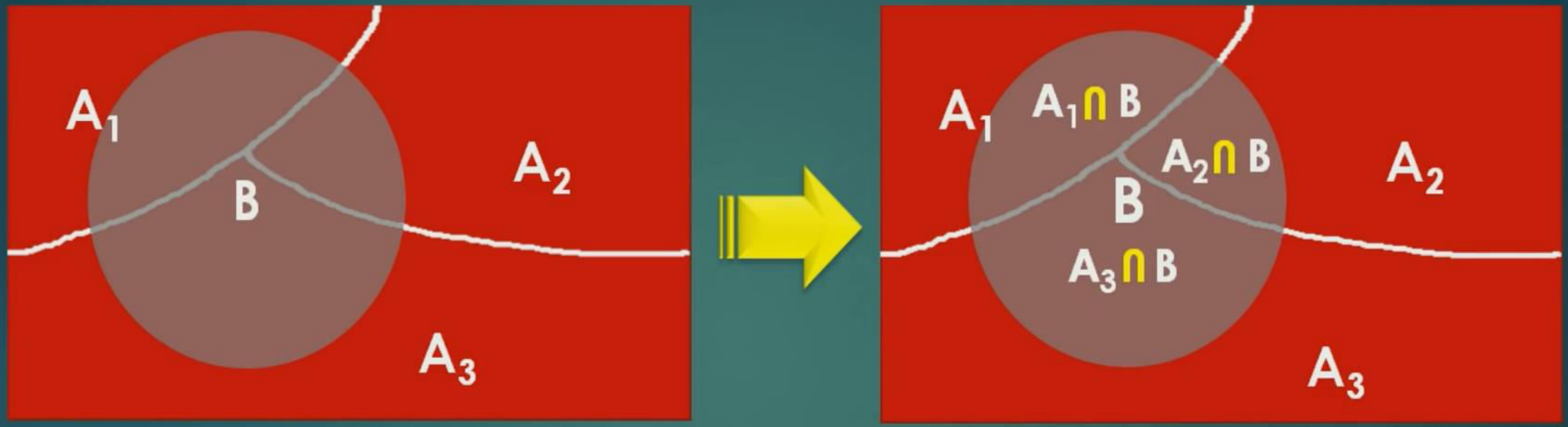
$$P(B \mid A) = \frac{P(A \cap B)}{P(A)}$$

$$P(A \cap B) = P(A \mid B) * P(B) = P(B \mid A) * P(A) \quad \text{---(i)}$$

$$P(A \mid B) * P(B) = P(B \mid A) * P(A)$$

$$P(A \mid B) = \frac{P(B \mid A) * P(A)}{P(B)} \quad \text{---(ii)}$$

Sample Space & Intersection of Events



$$P(B) = P(A_1 \cap B) + P(A_2 \cap B) + P(A_3 \cap B)$$

$$P(B) = P(B \mid A_1)*P(A_1) + P(B \mid A_2)*P(A_2) + P(B \mid A_3)*P(A_3)$$

$$P(B) = \sum_{i=1}^{n} P(B \mid A_i)*P(A_i) \quad \text{- - - (iii)}$$

Putting it all together

$$P(A_i \mid B) = \frac{P(B \mid A_i)*P(A_i)}{P(B)} \quad \text{- - - -(ii)}$$

$$P(B) = \sum_{i=1}^{n} P(B \mid A_i)*P(A_i) \quad \text{- - - -(iii)}$$

$$P(A_i \mid B) = \frac{P(B \mid A_i)*P(A_i)}{\sum_{i=1}^{n} P(B \mid A_i)*P(A_i)}$$

Bayes Theorem

- 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 is the cornerstone of Bayesian learning methods because it provides a way to calculate the posterior probability $P(h|D)$, from the prior probability $P(h)$, together with $P(D)$ and $P(D|h)$.

Bayes Theorem

Determining the best hypothesis from some space H, given the observed training data D

- **Bayes Theorem → to calculate $P(h/D)$**

$$P(h/D) = \frac{P(D/h) * P(h)}{P(D)}$$

Bayes Theorem

Determining the best hypothesis from some space H, given the observed training data D

- **Bayes Theorem → to calculate $P(h/D)$**

Probability of observing data D given some hypothesis h that holds

$$P(h/D) = \frac{P(D/h) * P(h)}{P(D)}$$

Bayes Theorem

Determining the best hypothesis from some space H, given the observed training data D

- **Bayes Theorem → to calculate $P(h/D)$**

Probability of observing data D given some hypothesis h that holds

Prior probability / initial probability that hypothesis h holds before observing the training data

$$P(h/D) = \frac{P(D/h) * P(h)}{P(D)}$$

Bayes Theorem

Determining the best hypothesis from some space H, given the observed training data D

- **Bayes Theorem → to calculate $P(h/D)$**

Probability of observing data D given some hypothesis h that holds

Prior probability / initial probability that hypothesis h holds before observing the training data

$$P(h/D) = \frac{P(D/h) * P(h)}{P(D)}$$

Prior probability of training data D (i.e. the probability of D given no knowledge about which hypothesis holds)

Bayes Theorem

Determining the best hypothesis from some space H, given the observed training data D

- **Bayes Theorem → to calculate $P(h/D)$**

Probability of observing data D given some hypothesis h that holds

Prior probability / initial probability that hypothesis h holds before observing the training data

$$P(h/D) = \frac{P(D/h) * P(h)}{P(D)}$$

Probability that h holds given the observed training data D.

Prior probability of training data D (i.e. the probability of D given no knowledge about which hypothesis holds)

Bayes Theorem

Determining the best hypothesis from some space H, given the observed training data D

- **Bayes Theorem → to calculate $P(h/D)$**

Probability of observing data D given some hypothesis h that holds

Prior probability / initial probability that hypothesis h holds before observing the training data

$$P(h/D) = \frac{P(D/h) * P(h)}{P(D)}$$

Probability that h holds given the observed training data D.

Prior probability of training data D (i.e. the probability of D given no knowledge about which hypothesis holds)

- $P(h/D)$ is called posterior probability of h, because it reflects our confidence that h holds after we have seen the training data D.

BAYES THEOREM

Notations

- **P(h)** *prior probability of* h, reflects any background knowledge about the chance that h is correct
- **P(D)** *prior probability of* D, probability that D will be observed
- **P(D|h)** probability of observing D given a world in which h holds
- **P(h|D)** *posterior probability of* h, reflects confidence that h holds after D has been observed

Choosing Hypothesis

$$P(h/D) = \frac{P(D/h) * P(h)}{P(D)}$$

- Find $h \in H$ which is most probable given the observed data
- Such maximally probable hypothesis is called ***maximum a posteriori*** (MAP) hypothesis

$$h_{MAP} \equiv \arg\max_{h \in H} P(h/D)$$

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

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

Choosing Hypothesis

$$P(h/D) = \frac{P(D/h) * P(h)}{P(D)}$$

- Find $h \in H$ which is most probable given the observed data
- Such maximally probable hypothesis is called ***maximum a posteriori*** (MAP) hypothesis

$$h_{MAP} \equiv \arg\max_{h \in H} P(h/D)$$

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

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

$P(D)$ can be omitted because it is a constant independent of h

Maximum likelihood hypothesis

- If we assume that every hypothesis in H is equally probable then

  $P(h_i) = P(h_j)$ for all h_i and h_j in H

- The most probable hypothesis h_{ML} is given as

$$h_{ML} \equiv \arg\max_{h \in H} P(D/h)$$

Maximum likelihood hypothesis

→ Hypothesis that maximizes $P(D/h)$

Likelihood of the Data *D* given *h*

SO FAR..

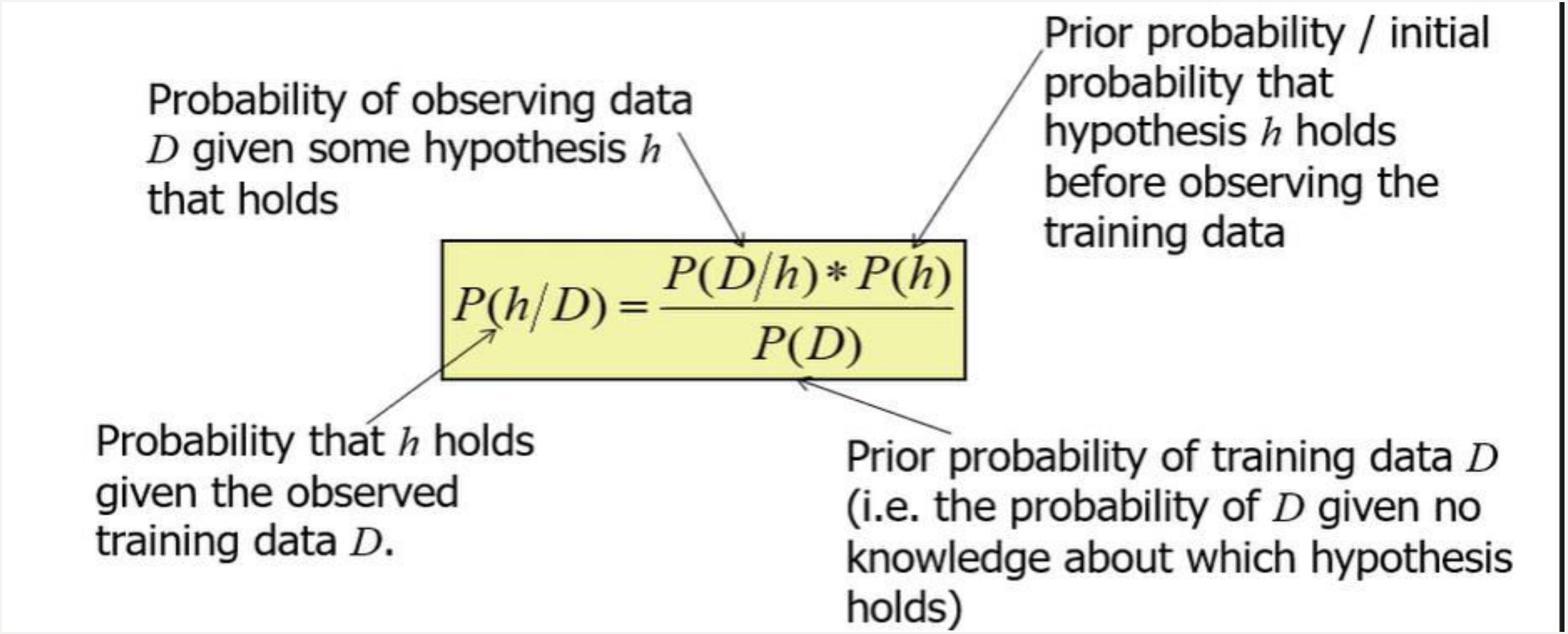
- Bayesian reasoning provides a probabilistic approach to inference.
- **Probability?**
- **Inference?**
- **Hypothesis?**
- **Prior Probability?**

SO FAR..

- 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 is the cornerstone of Bayesian learning methods because it provides a way to calculate the posterior probability $P(h|D)$, from
  - **the prior** probability $P(h)$,
  - **Probabilty over the data set** $P(D)$ and
  - **Current probability** $P(D|h)$.

SO FAR..


Probability of observing data D given some hypothesis h that holds
Prior probability / initial probability that hypothesis h holds before observing the training data
$P(h/D) = \frac{P(D/h) * P(h)}{P(D)}$
Probability that h holds given the observed training data D.
Prior probability of training data D (i.e. the probability of D given no knowledge about which hypothesis holds)

SO FAR..

Maximum A Posteriori (MAP) Hypothesis, h_{MAP}

- The learner considers some set of candidate hypotheses H and it is interested in finding the ***most probable hypothesis*** h ∈ H given the observed data D
- Any such maximally probable hypothesis is called a ***maximum a posteriori (MAP) hypothesis*** h_{MAP}.
- We can determine the MAP hypotheses by using Bayes theorem to calculate the posterior probability of each candidate hypothesis.

$$h_{MAP} \equiv \arg\max_{h \in H} P(h/D)$$

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

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

$P(D)$ can be omitted because it is a constant independent of h

Maximum Likelihood (ML) Hypothesis, h_{ML}

- If we assume that every hypothesis in H is equally probable

  i.e. $P(h_i) = P(h_j)$ for all h_i and h_j in H

  We can only consider P(D|h) to find the most probable hypothesis.

- P(D|h) is often called the ***likelihood*** of the data ***D*** given ***h***
- Any hypothesis that maximizes P(D|h) is called a ***maximum likelihood (ML) hypothesis, h_{ML}.***

$$h_{ML} \equiv \arg\max_{h \in H} P(D/h)$$

Maximum likelihood hypothesis
→ Hypothesis that maximizes $P(D/h)$

Likelihood of the Data *D* given *h*

Example

- A medical diagnosis problem with two alternative hypotheses
  - The patient has a particular form of cancer / does not have cancer

- Given, over the entire population of people only 0.008 have the disease.
- The lab test is only an imperfect indicator of disease. The test returns a correct positive result in only 98% of the cases in which the disease is actually present. It gives correct negative result in only 97% of the cases in which the disease is not present.
- **New patient is observed for whom lab test returns a positive result. Should we diagnose the patient as having cancer or not?**

Example

- A medical diagnosis problem with two alternative hypotheses
  - The patient has a particular form of cancer / does not have cancer
- Consider a medical diagnosis problem in which there are two alternative hypotheses

✓The patient has a particular form of cancer (denoted by *cancer*)

✓The patient does not (denoted by ¬ *cancer*)

- The available data is from a particular laboratory with two possible outcomes: +(positive) and - (negative)

EXAMPLE

- 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 cancer.

- $P(cancer) = 0.008$
- $P(\neg cancer) = 0.992$ *[1-P(Cancer)]*
- $P(+/cancer) = 0.98$
- $P(-/cancer) = 0.02$ [1- $P(-/cancer)$]
- $P(+/\neg cancer) = 0.03$ [1- $P(-/\neg cancer)$]
- $P(-/\neg cancer) = 0.97$
- New patient is observed for whom lab test returns a positive result. Should we diagnose the patient as having cancer or not?

$$P(cancer/+) = \frac{P(+/cancer) * P(cancer)}{P(+)} = \frac{0.98 * 0.008}{P(+)} = \frac{0.0078}{P(+)}$$

$$P(\neg cancer/+) = \frac{P(+/\neg cancer) * P(\neg cancer)}{P(+)} = \frac{0.0298}{P(+)}$$

Maximum

➔ $h_{MAP} = \neg cancer$

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 $A_1, \ldots, A_n$ are mutually exclusive with $\sum_{i=1}^{n} P(A_i) = 1$, then

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

Brute-Force Bayes Concept Learning

- A Concept-Learning algorithm considers a finite hypothesis space **H** defined over an instance space **X**
- The task is to learn the target concept (a function) **c : X → {0,1}.**
- The learner gets a set of training examples **(<x_1,d_l> . . . <x_m,d_m>)** where $\mathbf{x_i}$ is an instance from **X** and $\mathbf{d_i}$ is its target value (i.e. $c(x_i) = d_i$).

- *Brute-Force Bayes Concept Learning Algorithm* finds the maximum a posteriori hypothesis (h_{MAP}), based on Bayes theorem.

Brute-Force MAP Learning Algorithm

- We can design a straight forward 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

- This algorithm may require significant computation, because it applies Bayes theorem to each hypothesis in H to calculate $P(h|D)$.
  - While this is impractical for large hypothesis spaces,
  - The algorithm is still of interest because it provides a standard against which we may judge the performance of other concept learning algorithms.

Brute-Force MAP Learning Algorithm

- BF MAP learning algorithm must specify values for **P(h)** and **P(D|h).**
- Let us choose **P(h)** and **P(D|h)**them to be consistent with the following assumptions:

1. The training data D is noise free (i.e., $d_i = c(x_i)$).
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 MAP Learning Algorithm

- Given these assumptions, 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.***
- Furthermore, because we assume the target concept is contained in **H** we should require that these prior probabilities sum to 1.
- Together these constraints imply that we should choose

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

Brute-Force MAP Learning Algorithm

- What choice shall we make for **P(D|h)*?***
- ***P(D|h)*** is the probability of observing the target values $\mathbf{D = \langle d_1 \ldots d_m \rangle}$ for the fixed set of instances $\mathbf{\langle x_1 \ldots x_m \rangle}$.
- Since we assume noise-free training data, the probability of observing classification $\mathbf{d_i}$ given **h** is just 1 **if** $\mathbf{d_i = h(x_i)}$ and 0 **if** $\mathbf{d_i \neq h(x_i)}$. 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 MAP Learning Algorithm

- **Let us consider the** first step of this algorithm, which uses Bayes theorem to compute the posterior probability **P(h|D) of each hypothesis h given the observed training data D.**

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

- First consider the case where h is inconsistent with the training data D.
- We know that P(D|h) to be 0 when h is inconsistent with D, we have,

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

- The posterior probability of a hypothesis inconsistent with D is zero.

Brute-Force MAP Learning Algorithm

- Now consider the case where **h is consistent** with D.
- We know that P(Dlh) to be 1 when h is consistent with D, we have

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

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

$$= \frac{1 \cdot \frac{1}{|H|}}{\frac{|VS_{H,D}|}{|H|}}$$

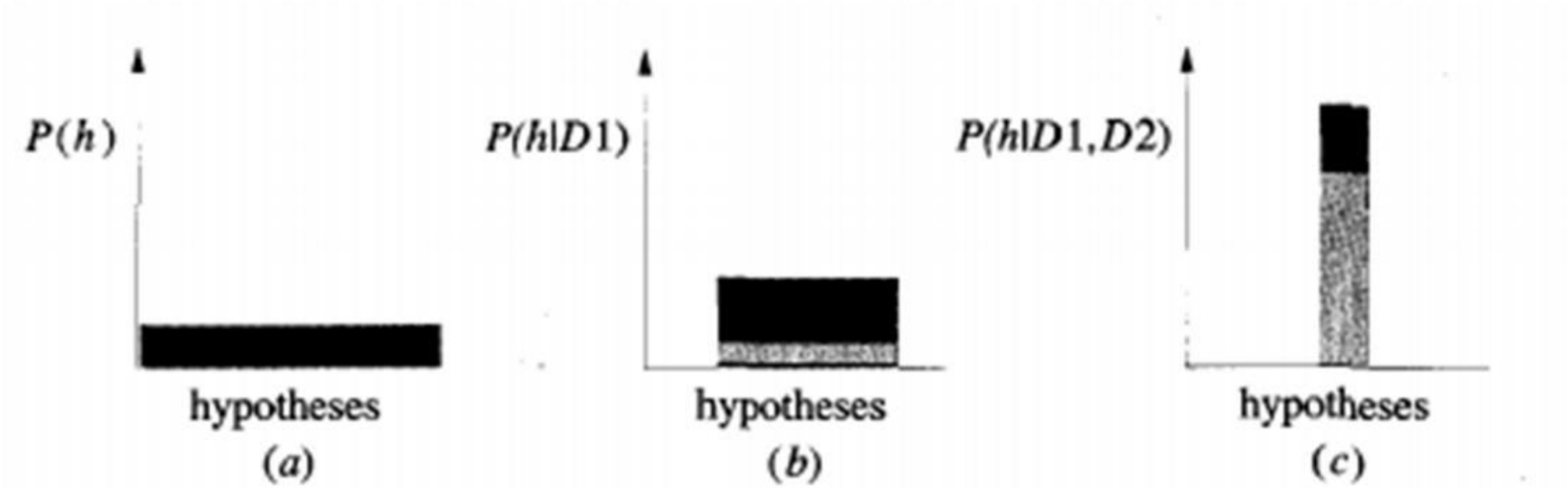
$$= \frac{1}{|VS_{H,D}|} \text{ if } h \text{ is consistent with } D$$

Brute-Force MAP Learning Algorithm

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

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

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



(a) Uniform priors assign equal probability to each hypothesis.
As training data increases first to Dl (b),then to Dl ∧ D2(c),
the posterior probability of inconsistent hypotheses becomes zero,
while posterior probabilities increase for hypotheses remaining in the version space.

MAP Hypotheses and Consistent Learners

- A learning algorithm is a ***consistent learner*** if 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 (i.e., $P(h_i) = P(h_j)$ for all i, j), and
  - deterministic, noise free training data (i.e., $P(D|h) = 1$ if D and h are consistent, and 0 otherwise).
- 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.
- Are there other probability distributions for P(h) and P(D|h) under which FIND-S outputs MAP hypotheses? Yes.
  - 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.

Maximum Likelihood and Least-Squared Error Hypotheses

- Many learning approaches such as neural network learning, linear regression, and polynomial curve fitting try to learn a continuous-valued target function.

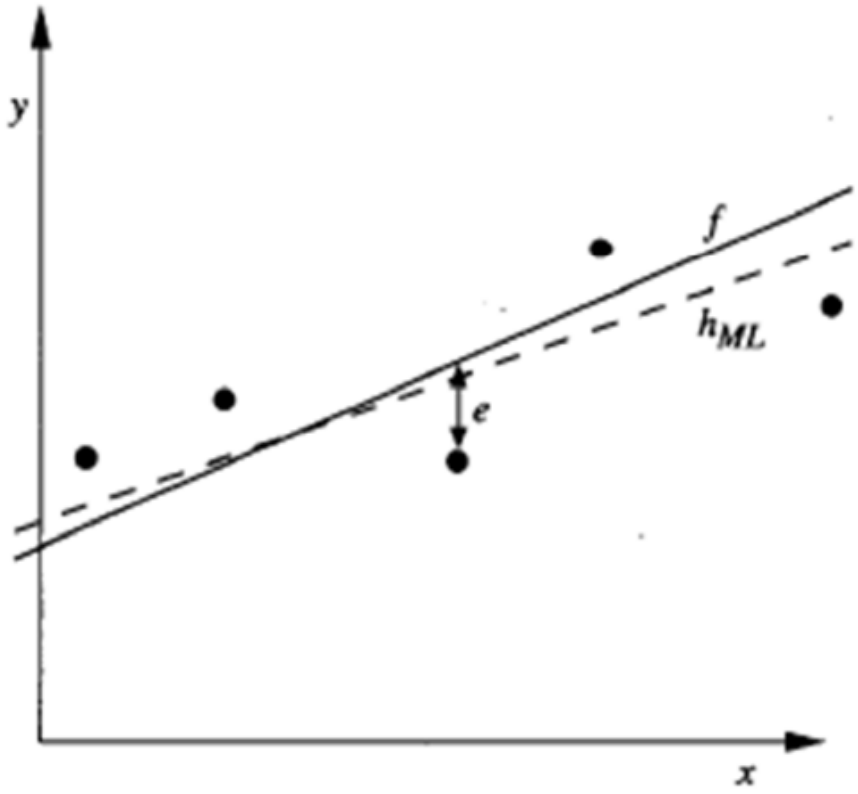
- **Under certain assumptions any learning algorithm that minimizes the squared error between the output hypothesis predictions and the training data will output a *MAXIMUM LIKELIHOOD HYPOTHESIS*.**

- The significance of this result is that it provides a Bayesian justification (under certain assumptions) for many neural network and other curve fitting methods that attempt to minimize the sum of squared errors over the training data.

Learning A Continuous-Valued Target Function

- 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 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
- Each training example is a pair of the form (x_i, d_i) where $d_i = f(x_i) + e_i$.
  - Here $f(x_i)$ is the noise-free value of the target function and e_i is a ***random variable*** representing the noise.
  - It is assumed that the values of the e_i 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.

Learning A Linear Function



- The target function f corresponds to the solid line.
- The training examples (x_i, d_i) 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 hypothesis h_{ML} with least-squared training error, hence the maximum likelihood hypothesis.
- Notice that the maximum likelihood hypothesis is not necessarily identical to the correct hypothesis, f, because it is inferred from only a limited sample of noisy training data.

Basic Concepts from Probability Theory

- Before showing why a hypothesis that minimizes the sum of squared errors in this setting is also a maximum likelihood hypothesis, let us quickly review basic concepts from probability theory

- Probability density: p for continuous variables
- Normal distribution: characterized by its mean μ and its standard deviation σ

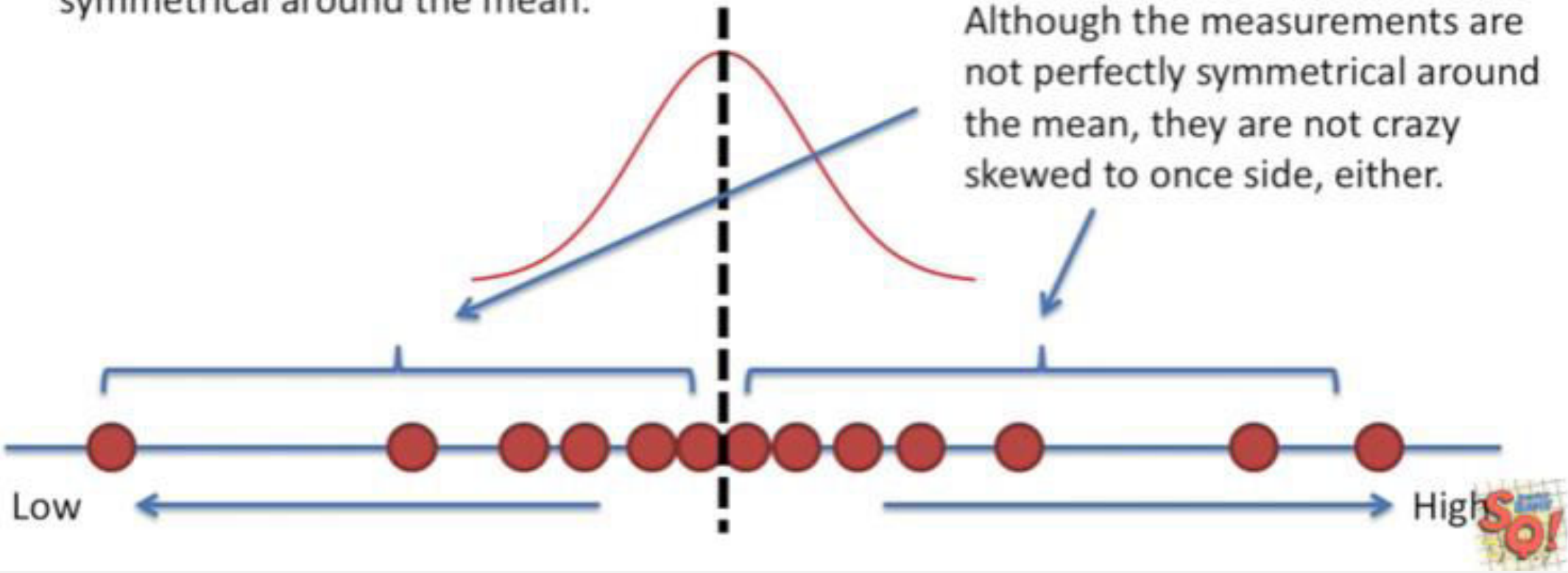
Probability density function:

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

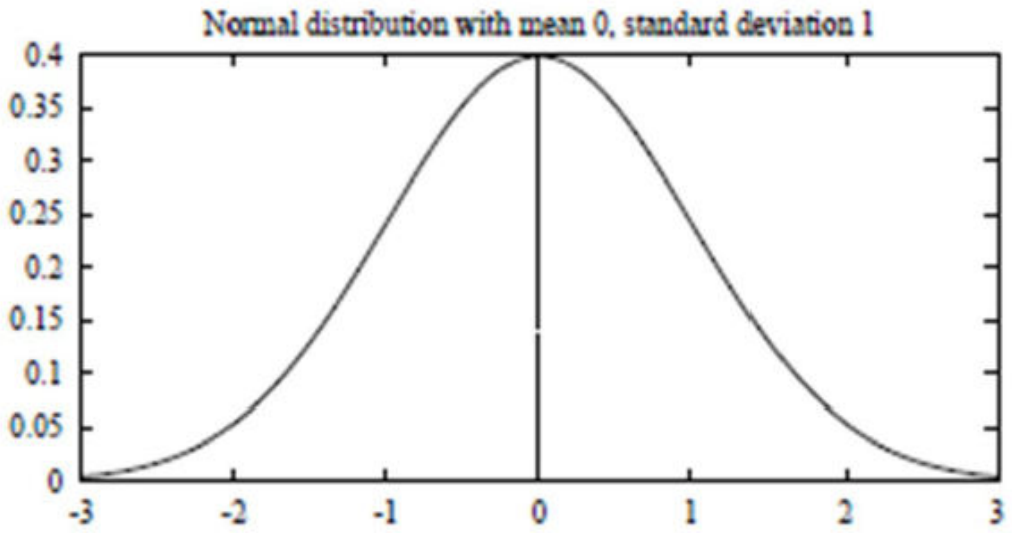
"Normally distributed" means a number of things:

1) We expect most of the measurements (mouse weights) to be close to the mean (average).

2) We expect the measurements to be relatively symmetrical around the mean.



Basic Concepts from Probability Theory



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

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

- A Normal distribution is fully determined by two parameters in the formula: μ and σ .
- If the random variable X follows a normal distribution:
  - The probability that X will fall into the interval (a, b) is $\int_a^b p(x)dx$
  - The expected, or ***mean value of X***, $E[X] = \mu$
  - The ***variance of X***, $\mathrm{Var}(X) = \sigma^2$
  - The ***standard deviation of X***, $\sigma_x = \sigma$

Maximum Likelihood and Least-Squared Error Hypotheses – Deriving h_{ML}

- In order to find the maximum likelihood hypothesis in Bayesian learning for continuous valued target function, we start with maximum likelihood hypothesis definition but using lower case ***p*** to refer to the probability density function.

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

- We assume a fixed set of training instances ($x_1 \ldots x_m$) and therefore consider the data D to be the corresponding sequence of target values $D = (d_1 \ldots d_m)$.
- Here we can write $p(D|h)$ as the product of the various $p(d_i|h)$

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

Maximum Likelihood and Least-Squared Error Hypotheses – Deriving h_{ML}

- Given that the noise e_i obeys a Normal distribution with zero mean and unknown variance σ^2, each d_i must also obey a Normal distribution with variance σ^2 centered around the true target value $f(x_i)$ rather than zero.
- $p(d_i|h)$ can be written as a Normal distribution with variance σ^2 and mean $\mu = f(x_i)$.
- Let us write the formula for this Normal distribution to describe $p(d_i|h)$, beginning with the general formula for a Normal distribution and substituting appropriate μ and σ^2 .
- Because we are writing the expression for the probability of d_i given that h is the correct description of the target function f, we will also substitute $\mu = f(xi) = h(xi)$,

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

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

Maximum Likelihood and Least-Squared Error Hypotheses – Deriving h_{ML}

- Maximizing **ln p** also maximizes **p**.

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

- First term is constant, discard it.

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

- Maximizing the negative quantity is equivalent to minimizing the corresponding positive quantity

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

- Finally, we can again discard constants that are independent of ***h.***

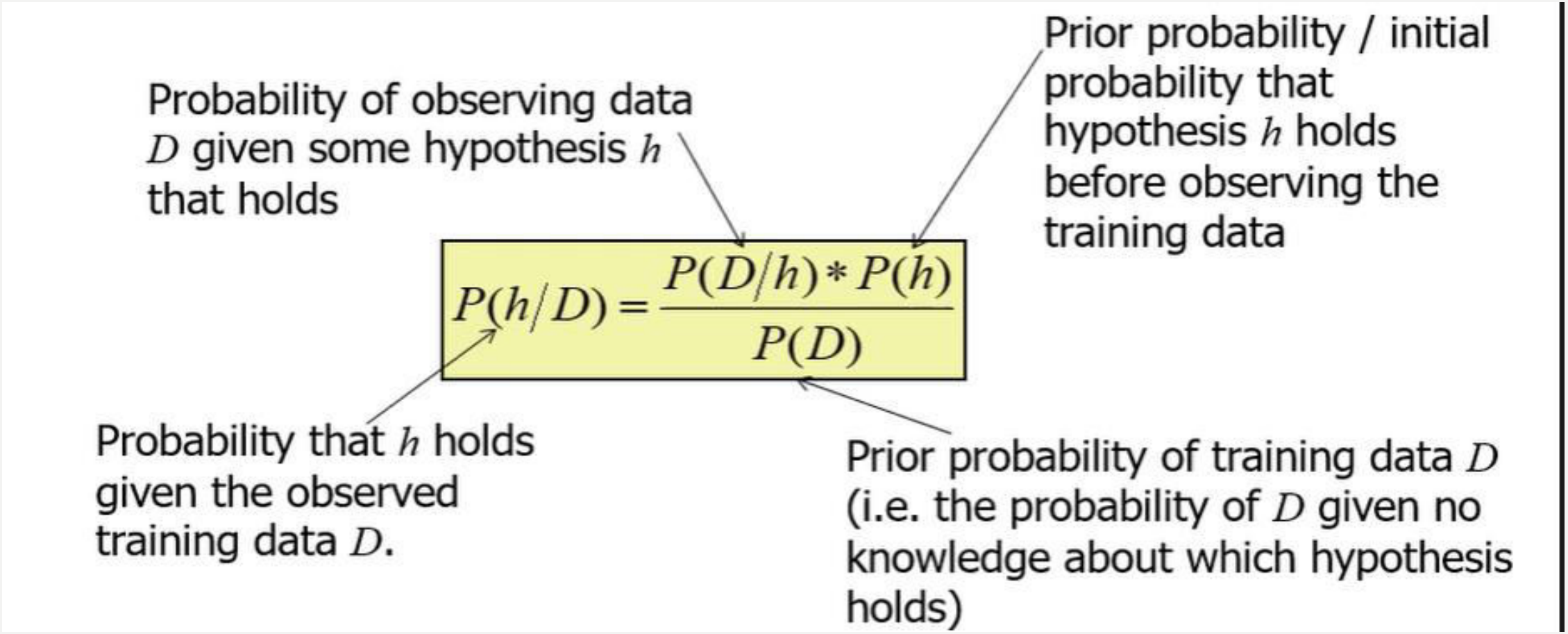
$$h_{ML} = \underset{h \in H}{\operatorname{argmin}} \sum_{i=1}^{m} (d_i - h(x_i))^2$$

Maximum Likelihood and Least-Squared Error Hypotheses

- The maximum likelihood hypothesis h_{ML} is the one that minimizes the sum of the squared errors between observed training values d_i and hypothesis predictions $h(x_i)$.
- This holds under the assumption that the observed training values d_i are generated by adding random noise to the true target value, where this random noise is drawn independently for each example from a Normal distribution with zero mean.
- Similar derivations can be performed starting with other assumed noise distributions, producing different results.
- Why is it reasonable to choose the Normal distribution to characterize noise?
  - One reason, is that it allows for a mathematically straightforward analysis.
  - A second reason is that the smooth, bell-shaped distribution is a good approximation to many types of noise in physical systems.
- Minimizing the sum of squared errors is a common approach in many neural network, curve fitting, and other approaches to approximating real-valued functions.

SO FAR..

Bayes Theorem



SO FAR..

Maximum A Posteriori (MAP) Hypothesis, h_{MAP}

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

Maximum Likelihood (ML) Hypothesis, h_{ML}

$$h_{ML} \equiv \arg\max_{h \in H} P(D/h)$$

Likelihood of the Data *D* given *h*

Brute-Force MAP Learning Algorithm

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

- BF MAP learning algorithm must specify values for **P(h)** and **P(D|h).**
- Let us choose **P(h)** and **P(D|h)**them to be consistent with the following assumptions:

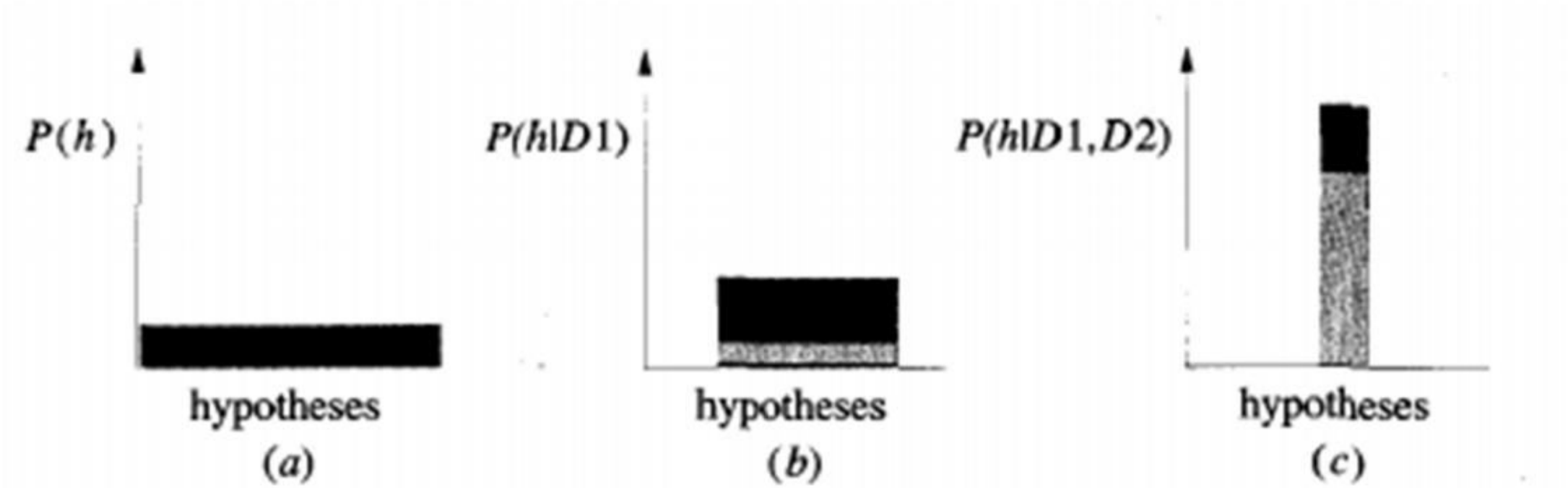
1. The training data D is noise free (i.e., $d_i = c(x_i)$).
2. The target concept c is contained in the hypothesis space H.
3. We have no a priori reason to believe that any hypothesis is more probable than any other.

Brute-Force MAP Learning Algorithm

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

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

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



(a) Uniform priors assign equal probability to each hypothesis.
As training data increases first to Dl (b),then to Dl ∧ D2(c),
the posterior probability of inconsistent hypotheses becomes zero,
while posterior probabilities increase for hypotheses remaining in the version space.

Maximum Likelihood and Least-Squared Error Hypotheses

- Many learning approaches such as neural network learning, linear regression, and polynomial curve fitting try to learn a continuous-valued target function.

- **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*.**

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

Maximum Likelihood Hypotheses for Predicting Probabilities

$$h_{ML} \equiv \arg\max_{h \in H} P(D/h)$$

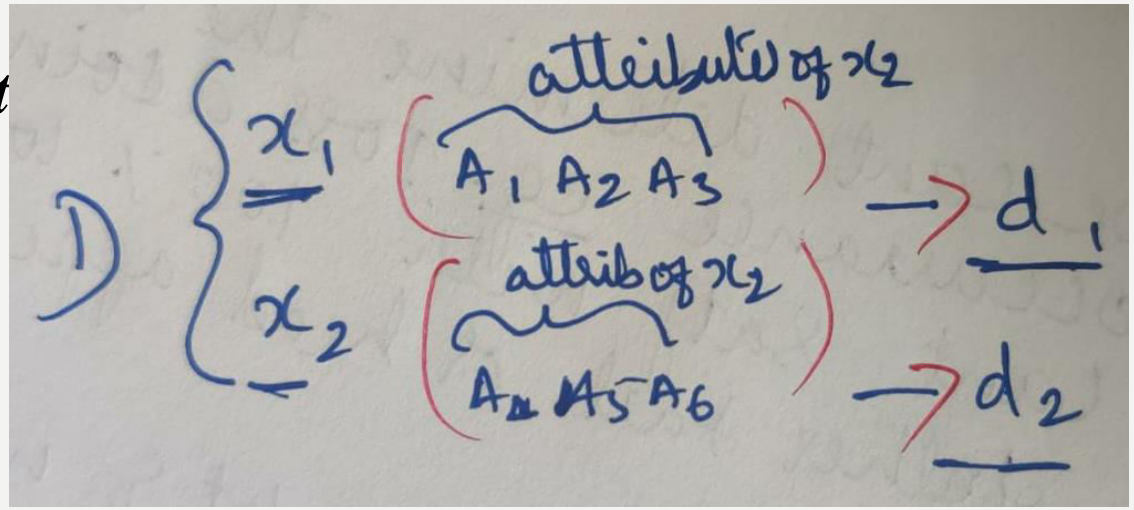
- Consider the setting in which we wish to learn a nondeterministic (probabilistic) function **f : X → {0, 1},** which has two discrete output values.
- For example, among a collection of patients exhibiting the same set of observable symptoms, we might find that 92% survive, and 8% do not.
- This unpredictability could arise from our inability to observe all the important distinguishing features of the patients.
- Given this problem setting, we might wish to learn a neural network (or other real-valued function approximator) whose output is the ***probability that f(x) = 1.***
- In other words , learn the target function

f ′ : X → [0, 1] such that **f ′ (x) = P(f(x) = 1)**

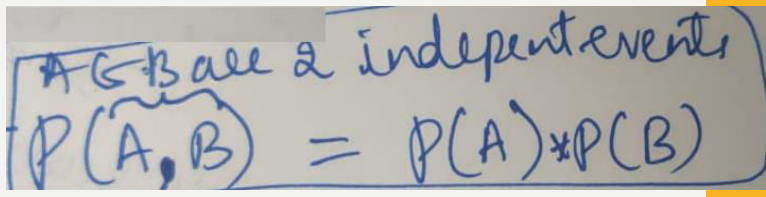
- In the above medical patient example, if x is one of those indistinguishable patients of which 92% survive, then f'(x) = 0.92 whereas the probabilistic function ***f (x) will be equal to 1 in 92% of cases and equal to 0 in*** the remaining 8%.

Maximum Likelihood Hypotheses for Predicting Probabilities

What criterion should we optimize in order t
for f ' in this setting?



- First obtain an expression for **P(D|h)**
- Assume the training data D is of the form observed 0 or 1 value for **f** ($\mathbf{x_i}$).
- Both x_i and d_i as random variables, and assuming that each training example is drawn independently, we can write **P(D|h)** as



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

- When **x** is independent of **h**, Applying the product rule

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

$$P(h/D) = \frac{P(D/h) * P(h)}{P(D)}$$

Maximum Likelihood Hypotheses for Predicting Probabilities

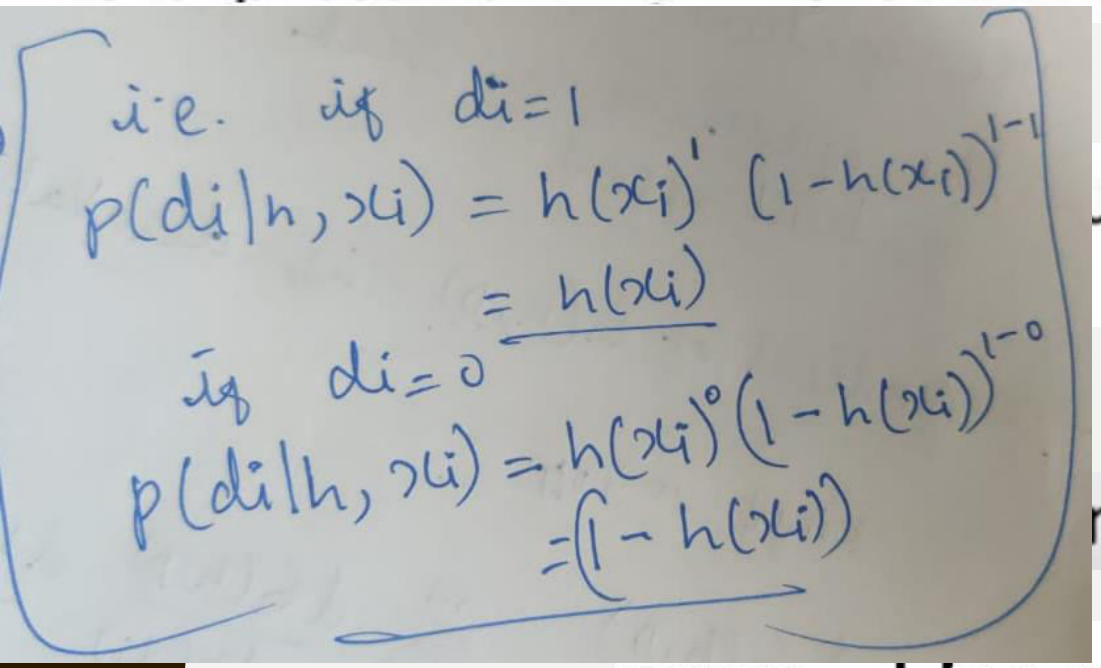
- Now what is the probability $P(d_i | h, x_i)$ of observing $d_i = 1$ for a single instance x_i, given a world in which hypothesis h holds?
- h is our hypothesis regarding the target function, which computes this very probability.
- Therefore, $P(d_i = 1 | h, x_i) = h(x_i)$, and in general

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

Maximum Likelihood Hypotheses for Predicting Probabilities

$$P(d_i|h, x_i) = \begin{cases} h(x_i) & \text{if } d_i = 1 \\ (1 - h(x_i)) & \text{if } d_i = 0 \end{cases} \qquad \text{equ (3)}$$

- In order to substitute this into the Equation (1) for **P(D|h)**, let us first re-express it in a more mathematically manipulatable form, as



$$P(d_i|h, x_i) = h(x_i)^{d_i}(1 - h(x_i))^{1-d_i} \qquad \text{equ (4)}$$

uation to substitute for **$P(d_i|h, x_i)$**

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

n for the maximum likelihood hypothesis

$$P(D|h) = \prod_{i=1} \prod h(x_i)^{d_i}(1 - h(x_i))^{1-d_i} P(x_i) \qquad \text{equ (5)}$$

Maximum Likelihood Hypotheses for Predicting Probabilities

- We write an expression for the maximum likelihood hypothesis

$$P(D|h) = \prod_{i=1}^{m} h(x_i)^{d_i} (1 - h(x_i))^{1-d_i} P(x_i) \qquad \text{equ (5)}$$

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

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} h(x_i)^{d_i} (1 - h(x_i))^{1-d_i} \qquad \text{equ (6)}$$

- The expression on the right side of Equation can be of the Binomial distribution

$$\log a^x = x. \log a$$

$$\log a.b = \log a + \log b$$

- As in earlier cases, we will find it easier to work with the lo

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} d_i \ln h(x_i) + (1 - d_i) \ln(1 - h(x_i)) \qquad \text{equ (7)}$$

- Equation (7) describes the quantity that must be maximized in order to obtain the maximum likelihood hypothesis in our current problem setting

Gradient Search To Maximize Likelihood In A Neural Net

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

- The gradient of G(h, D) is given by the vector of partial derivatives of G(h, D) with respect to the various network weights that define the hypothesis h represented by the learned network
- In this case, the partial derivative of G(h, D) with respect to weight w_{jk} from input k to unit j is

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

$$= \sum_{i=1}^{m} \frac{\partial (d_i \ln h(x_i) + (1 - d_i) \ln(1 - h(x_i)))}{\partial h(x_i)} \frac{\partial h(x_i)}{\partial w_{jk}}$$

$$= \sum_{i=1}^{m} \frac{d_i - h(x_i)}{h(x_i)(1 - h(x_i))} \frac{\partial h(x_i)}{\partial w_{jk}} \qquad \text{equ (1)}$$

Gradient Search To Maximize Likelihood In A Neural Net

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

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

- where x_{ijk} is the k^{th} input to unit j for the i^{th} training example, and d(x) is the derivative of the sigmoid squashing function.
- Finally, substituting this expression into Equation (1), we obtain a simple expression for the derivatives that constitute the gradient

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

Gradient Search To Maximize Likelihood In A Neural Net

➢Because we seek to maximize rather than minimize $\mathbf{P(D|h)}$, we perform ***gradient ascent*** rather than ***gradient descent search***. On each iteration of the search the weight vector is adjusted in the direction of the gradient, using the weight update rule

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

Where,

$$\Delta w_{jk} = \alpha \sum_{i=1}^{m} (d_i - h(x_i))\ x_{ijk} \qquad \text{equ (2)}$$

where η is a small positive constant that determines the step size of the i gradient ascent search

➢It is interesting to compare this weight-update rule to the weight-update rule used by the BACKPROPAGATION algorithm to minimize the sum of squared errors between predicted and observed network outputs.

➢The BACKPROPAGATION update rule for output unit weights, re-expressed using our current notation, is

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

Where,

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

Minimum Description Length Principle

- We have already learnt about **Occam's razor,** a popular inductive bias that can be summarized as **"choose the shortest explanation for the observed data.“**
- Here, we consider Bayesian perspective on Occam’s razor a closely related principle called the **Minimum Description Length (MDL)** principle.
- The Minimum Description Length (MDL) principle is motivated by interpreting the definition **of h_{MAP}** in the light of basic concepts from information theory.

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

$$\log a.b = \log a + \log b$$

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

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

or alternatively, minimizing the negative of this quantity

$$h_{MAP} = \underset{h \in H}{argmin}\ -\log_2 P(D|h) - \log_2 P(h) \qquad \text{equ (1)}$$

- This equation can be interpreted as a statement that short hypotheses are preferred.

Minimum Description Length Principle

- Consider the problem of designing a code to transmit messages drawn at random
- If i is the message, The probability of encountering message **i** is $\mathbf{p_i}$
- Interested in the most compact code; that is, interested in the code that minimizes the expected number of bits we must transmit in order to encode a message drawn at random.
- To minimize the expected code length we should assign shorter codes to messages that are more probable.
- Shannon and Weaver (1949) showed that the optimal code (i.e., the code that minimizes the expected message length) assigns $\mathbf{-\log_2 p_i}$ bits to encode message i.
- The number of bits required to encode message i using code C as the ***description length of message i with respect to C***, which we denote by $L_c(i)$.

Minimum Description Length Principle

Interpreting the equation

$$h_{MAP} = \underset{h \in H}{argmin} \; -\log_2 P(D|h) - \log_2 P(h) \qquad \text{equ (1)}$$

- **$-\log_2 P(h)$:** the description length of h under the optimal encoding for the hypothesis space H $\mathbf{L_{C_H}(h) = -\log_2 P(h)}$, where C_H is the optimal code for hypothesis space H.
- **$-\log_2 P(D \mid h)$:** the description length of the training data D given hypothesis h, under the optimal encoding fro the hypothesis space H: $\mathbf{L_{C_{D|h}}(D|h) = -\log_2 P(D|\, h)}$, where $C_{D|h}$ is the optimal code for describing data D assuming that both the sender and receiver know the hypothesis h.

- Rewrite Equation (1) to show that h_{MAP} is the hypothesis h that minimizes the sum given by the description length of the hypothesis plus the description length of the data given the hypothesis.

$$h_{MAP} = \underset{h \in H}{argmin} \; L_{C_H}(h) + L_{C_{D|h}}(D|h)$$

where C_H and $C_{D|h}$ are the optimal encodings for H and for D given h

Minimum Description Length Principle

- The minimum description length (MDL) principle recommends choosing the hypothesis that minimizes the sum of these two description lengths of equation.

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

Minimum Description Length principle:

$$h_{MDL} = \underset{h \in H}{\text{argmin}}\, L_{C_1}(h) + L_{C_1}(D \mid h)$$

- Where, codes C_1 and C_2 to represent the hypothesis and the data given the hypothesis
- The above analysis shows that if we choose C_1 to be the optimal encoding of hypotheses C_H, and if we choose C_2 to be the optimal encoding $C_{D|h}$, then $\mathbf{h_{MDL} = h_{MAP}}$.

Application To Decision Tree Learning

- Apply the MDL principle to the problem of learning decision trees from some training data.

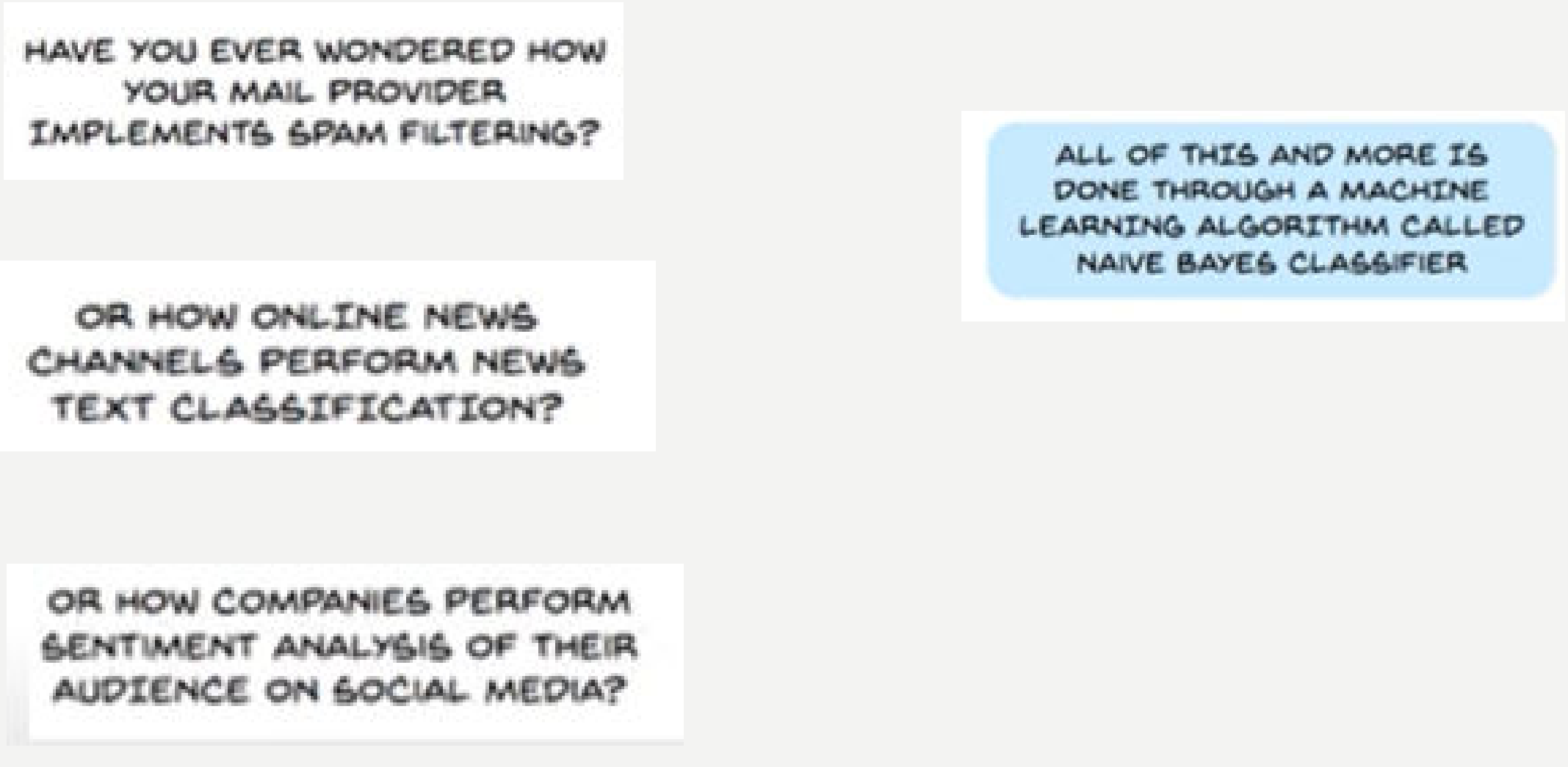
What should we choose for the representations C_1 and C_2 of hypotheses and data?

- **For C_1:** C_1 might be some obvious encoding, in which the description length grows with the number of nodes and with the number of edges
- **For C_2:** Suppose that the sequence of instances (x_1 . . .x_m) is already known to both the transmitter and receiver, so that we need only transmit the classifications (f (x_1)…f(x_m)).
- Now if the training classifications (f (x_1) . . .f(x_m)) are identical to the predictions of the hypothesis, then there is no need to transmit any information about these examples. The description length of the classifications given the hypothesis ZERO
- If examples are misclassified by h, then for each misclassification we need to transmit a message that identifies which example is misclassified as well as its correct classification
- The hypothesis $\mathbf{h_{MDL}}$ under the encoding C_1 and C_2 is just the one that minimizes the sum of these description lengths.

Minimum Description Length Principle

- Thus the MDL principle provides a way of trading off hypothesis complexity for the number of errors committed by the hypothesis.
- It might select a shorter hypothesis that makes a few errors over a longer hypothesis that perfectly classifies the training data.
- MDL provides a way to deal with the issue of overfitting the data.

NAVIE BAYES CLASSIFIER



NAVIE BAYES CLASSIFIER

Naive Bayes Classifier is based on Bayes' Theorem which gives the conditional probability of an event A given B

Bayes Theorem

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

where:

P(A|B) = Conditional Probability of A given B

P(B|A) = Conditional Probability of A given B

P(A) = Probability of event A

P(B) = Probability of event A

Probability of observing data D given some hypothesis h that holds

Prior probability / initial probability that hypothesis h holds before observing the training data

$$P(h/D) = \frac{P(D/h) * P(h)}{P(D)}$$

Probability that h holds given the observed training data D.

Prior probability of training data D (i.e. the probability of D given no knowledge about which hypothesis holds)

NAVIE BAYES CLASSIFIER

- One highly practical Bayesian learning method is the naive Bayes learner, often called the ***naive bayes classifier***
- In some domains its performance has been shown to be comparable to that of neural network and decision tree learning

- The naive Bayes classifier applies to learning tasks where each instance x is described by a conjunction of attribute values and where the target function $f(x)$ can take on any value from some finite set V
- A set of training examples of the target function is provided, and a new instance is presented, described by the tuple of attribute values $\langle a_1, a_2, \ldots, a_n \rangle$
- The learner is asked to predict the target value, or classification, for this new instance.

NAVIE BAYES CLASSIFIER

- The Bayesian approach to classifying the new instance is to assign the most probable target value, v_{MAP}, given the attribute values $<a_1, a_2, \ldots, a_n>$ that describe the instance

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

$$P(h/D) = \frac{P(D/h) * P(h)}{P(D)}$$

We can use Bayes theorem to rewrite this expression as

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

$$= \underset{v_j \in V}{\operatorname{argmax}} P(a_1, a_2 \ldots a_n | v_j) P(v_j) \quad \text{--------} \quad (1)$$

NAVIE BAYES CLASSIFIER

- Now we could attempt to estimate the two terms in the above Equation based on the training data
- 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
- The naive Bayes classifier is based on the simplifying assumption that the attribute values are conditionally independent given the target value
- The probability of observing the conjunction $a_1, a_2, \ldots, a_n$ is just the product of the probabilities for the individual attributes: $P(a_1, a_2, \ldots, a_n | v_j) = \prod_i P(a_i | v_j)$

NAVIE BAYES CLASSIFIER

➢Substituting this into Equation **(1), we have the approach used by the naive Bayes classifier.**

Naive Bayes classifier:

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

where $\mathbf{V_{NB}}$ denotes the target value output by the naive Bayes classifier.

➢To summarize, the naive Bayes learning method involves a learning step in which the various $P(v_j)$ and $P(a_i|v_j)$ terms are estimated, based on their frequencies over the training data.

➢The set of these estimates corresponds to the learned hypothesis.

➢This hypothesis is then used to classify each new instance by applying the rule in Equation.

An Illustrative Example

- Let us apply the naive Bayes classifier to a concept learning problem we considered before: classifying days according to whether someone will play tennis

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
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	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	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

An Illustrative Example

- Here we use the naive Bayes classifier and the training data from this table to classify the following novel instance:

$$\langle Outlook = sunny, Temperature = cool, Humidity = high, Wind = strong \rangle$$

- Our task is to predict the target value ***(yes or no)*** of the target concept ***PlayTennis*** for this new instance
- To fit the current task, the target value v_{NB} is given by

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

$$= \underset{v_j \in \{yes,no\}}{\text{argmax}} P(v_j) \quad P(Outlook = sunny|v_j) P(Temperature = cool|v_j)$$

$$\cdot P(Humidity = high|v_j) P(Wind = strong|v_j)$$

NAIVE BAYES CLASSIFIER – Example -1

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
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	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	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

$\langle Outlook = sunny, Temperature = cool, Humidity = high, Wind = strong \rangle$

NAIVE BAYES CLASSIFIER – Example -1

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
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	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	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

$\langle Outlook = sunny, Temperature = cool, Humidity = high, Wind = strong \rangle$

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
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	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	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

$$P(PlayTennis = yes) = 9/14 = .64$$

$$P(PlayTennis = no) = 5/14 = .36$$

NAIVE BAYES CLASSIFIER

Example - 1

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	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

$$P(PlayTennis = yes) = 9/14 = .64$$

$$P(PlayTennis = no) = 5/14 = .36$$

NAIVE BAYES CLASSIFIER

Example - 1

Outlook	Y	N
sunny	2/9	3/5
overcast	4/9	0
rain	3/9	2/5
Tempreature		
hot	2/9	2/5
mild	4/9	2/5
cool	3/9	1/5

Humidity	Y	N
high	3/9	4/5
normal	6/9	1/5
Windy		
Strong	3/9	3/5
Weak	6/9	2/5

NAIVE BAYES CLASSIFIER – Example -1

$$\langle Outlook = sunny, Temperature = cool, Humidity = high, Wind = strong \rangle$$

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

$$= \underset{v_j \in \{yes,no\}}{\text{argmax}} P(v_j) \quad P(Outlook = sunny|v_j) P(Temperature = cool|v_j)$$

$$\cdot P(Humidity = high|v_j) P(Wind = strong|v_j)$$

$$v_{NB}(yes) = P(yes)\ P(sunny|yes)\ P(cool|yes)\ P(high|yes)\ P(strong|yes) = .0053$$

$$=0.64*2/9*3/9*3/9*3/9=0.0053$$

$$v_{NB}(no) = P(no)\ P(sunny|no)\ P(cool|no)\ P(high|no)\ P(strong|no) = .0206$$

$$=0.36*3/5*1/5*4/5*3/5=0.206$$

$$v_{NB}(yes) = \frac{v_{NB}(yes)}{v_{NB}(yes)+v_{NB}(no)} = 0.205 \qquad v_{NB}(no) = \frac{v_{NB}(no)}{v_{NB}(yes)+v_{NB}(no)} = 0.795$$

NAIVE BAYES CLASSIFIER – Example -1

$$\langle Outlook = sunny, Temperature = cool, Humidity = high, Wind = strong \rangle$$

- 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.

NAIVE BAYES CLASSIFIER Example – 2

- Estimate conditional probabilities of each attributes {color, legs, height, smelly} for species classes: {M, H} using the data given in the table.
- Using these probabilities estimate the probability values for the new instance – (Color=Green, legs=2, Height=Tall, and Smelly=No).

No	Color	Legs	Height	Smelly	Species
1	White	3	Short	Yes	M
2	Green	2	Tall	No	M
3	Green	3	Short	Yes	M
4	White	3	Short	Yes	M
5	Green	2	Short	No	H
6	White	2	Tall	No	H
7	White	2	Tall	No	H
8	White	2	Short	Yes	H

NAIVE BAYES CLASSIFIER

EXAMPLE - 2

No	Color	Legs	Height	Smelly	Species
1	White	3	Short	Yes	M
2	Green	2	Tall	No	M
3	Green	3	Short	Yes	M
4	White	3	Short	Yes	M
5	Green	2	Short	No	H
6	White	2	Tall	No	H
7	White	2	Tall	No	H
8	White	2	Short	Yes	H

New Instance

(Color=Green, legs=2, Height=Tall, and Smelly=No)

$$P(M) = \frac{4}{8} = 0.5 \quad P(H) = \frac{4}{8} = 0.5$$

Color	M	H
White	2/4	3/4
Green	2/4	1/4

Legs	M	H
2	1/4	4/4
3	3/4	0/4

Height	M	H
Tall	3/4	2/4
Short	1/4	2/4

Smelly	M	H
Yes	3/4	1/4
No	1/4	3/4

NAIVE BAYES CLASSIFIER - EXAMPLE - 2

$$P(M) = \frac{4}{8} = 0.5 \quad P(H) = \frac{4}{8} = 0.5$$

Color	M	H
White	2/4	3/4
Green	2/4	1/4

Legs	M	H
2	1/4	4/4
3	3/4	0/4

Height	M	H
Tall	3/4	2/4
Short	1/4	2/4

Smelly	M	H
Yes	3/4	1/4
No	1/4	3/4

$$p(M|New\ Instance) = p(M) * p(Color = Green|M) * p(Legs = 2|M) * p(Height = tall|M) * p(Smelly = no\ |M)$$

$$p(M|New\ Instance) = 0.5 * \frac{2}{4} * \frac{1}{4} * \frac{3}{4} * \frac{1}{4} = 0.0117$$

$$p(H|New\ Instance) = p(H) * p(Color = Green|H) * p(Legs = 2|H) * p(Height = tall|H) * p(Smelly = no\ |H)$$

$$p(H|New\ Instance) = 0.5 * \frac{1}{4} * \frac{4}{4} * \frac{2}{4} * \frac{3}{4} = 0.047$$

$$p(H|New\ Instance) > p(M|New\ Instance)$$

Hence the new instance belongs to Speices H

NAIVE BAYES CLASSIFIER – Example – 3

Example No.	Color	Type	Origin	Stolen?
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	Yes
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV	Domestic	No
9	Red	SUV	Imported	No
10	Red	Sports	Imported	Yes

$$\textit{New Instance} = (\textit{Red}, \textit{SUV}, \textit{Domestic}) \rightarrow (\textit{Yes or No})$$

NAIVE BAYES CLASSIFIER

EXAMPLE - 3

Example No.	Color	Type	Origin	Stolen?
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	Yes
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV	Domestic	No
9	Red	SUV	Imported	No
10	Red	Sports	Imported	Yes

$$New\ Instance = (Red, SUV, Domestic)$$

$$p(Yes) = \frac{5}{10} = 0.5$$

$$p(No) = \frac{5}{10} = 0.5$$

Color	Yes	No
Red	3/5	2/5
Yellow	2/5	3/5

Type	Yes	No
Sports	4/5	2/5
SUV	1/5	3/5

Origin	Yes	No
Domestic	2/5	3/5
Imported	3/5	2/5

$$P(Yes|New\ Instance) = p(Yes) * P(Color = Red|Yes) * P(Type = SUV|Yes) * P(Origin = Domestic|Yes)$$

$$P(Yes|New\ Instance) = \frac{5}{10} * \frac{3}{5} * \frac{1}{5} * \frac{2}{5} = \frac{3}{125} = 0.024$$

$$P(No|New\ Instance) = p(No) * P(Color = Red|No) * P(Type = SUV|No) * P(Origin = Domestic|No)$$

$$P(No|New\ Instance) = \frac{5}{10} * \frac{2}{5} * \frac{3}{5} * \frac{3}{5} = \frac{9}{125} = 0.072$$

$$P(No|New\ Instance) > P(Yes|New\ Instance)$$

No

Example: The Art Competition has entries from three painters: Pam, Pia and Pablo

- Pam put in 15 paintings, 4% of her works have won First Prize.
- Pia put in 5 paintings, 6% of her works have won First Prize.
- Pablo put in 10 paintings, 3% of his works have won First Prize.

What is the chance that Pam will win First Prize?

- Pam put in 15 paintings, 4% of her works have won First Prize.
- Pia put in 5 paintings, 6% of her works have won First Prize.
- Pablo put in 10 paintings, 3% of his works have won First Prize.

$$P(\text{Pam}|\text{First}) = \frac{P(\text{Pam})P(\text{First}|\text{Pam})}{P(\text{Pam})P(\text{First}|\text{Pam}) + P(\text{Pia})P(\text{First}|\text{Pia}) + P(\text{Pablo})P(\text{First}|\text{Pablo})}$$

Put in the values:

$$P(\text{Pam}|\text{First}) = \frac{(15/30) \times 4\%}{(15/30) \times 4\% + (5/30) \times 6\% + (10/30) \times 3\%}$$

Multiply all by 30 (makes calculation easier):

$$P(\text{Pam}|\text{First}) = \frac{15 \times 4\%}{15 \times 4\% + 5 \times 6\% + 10 \times 3\%}$$

$$= \frac{0.6}{0.6 + 0.3 + 0.3}$$

$$= 50\%$$

A good chance!

Pam isn't the most successful artist, but she did put in lots of entries.

NAIVE BAYES CLASSIFIER EXAMPLE

Consider a football game between two rival teams, say team A and team B. Suppose team A wins 65% of the time and team B wins the remaining matches. Among the games won by team A, only 35% of them comes from playing at team B's football field. On the other hand, 75% of the victories for team B are obtained while playing at home.

1. If team B is to host the next match between the two teams, what is the probability that it will emerge as the winner?
2. If team B is to host the next match between the two teams, who will emerge as the winner?

Solution:

Y – Winning football match

X – Hosting football match

Probability that team A wins is $P(Y_A) = 0.65$.

Probability that team B wins is $P(Y_B) = 1 - P(Y_A) = 0.35$

Probability that team B hosted the match it had won is $P(X_B|Y_B) = 0.75$.

Probability that team B hosted the match won by team A is $P(X_B|Y_A) = 0.35$.

1. If team B is to host the next match between the two teams, what is the probability that it will emerge as the winner?

Solution:

Probability that team A wins is $P(Y_A) = 0.65$.

Probability that team B wins is $P(Y_B) = 1 - P(Y_A) = 0.35$

Probability that team B hosted the match it had won is $P(X_B|Y_B) = 0.75$.

Probability that team B hosted the match won by team A is $P(X_B|Y_A) = 0.35$.

$$P(Y_B|X_B) = \frac{P(X_B|Y_B) \times P(Y_B)}{P(X_B)}$$

$$= \frac{P(X_B|Y_B) \times P(Y_B)}{P(X_B|Y_A)P(Y_A) + P(X_B|Y_B)P(Y_B)}$$

$$= \frac{0.75 \times 0.35}{(0.35 \times 0.65 + 0.75 \times 0.35)}$$

$$= 0.5357$$

2. If team B is to host the next match between the two teams, who will emerge as the winner?

Solution:

Probability that team A wins is $P(Y_A) = 0.65$.

Probability that team B wins is $P(Y_B) = 1 - P(Y_A) = 0.35$

$$P(Y_A|X_B) = \frac{P(X_B|Y_A) \times P(Y_A)}{P(X_B)}$$

Probability that team B hosted the match it had won is $P(X_B|Y_B) = 0.75$.

$$= \frac{P(X_B|Y_A) \times P(Y_A)}{P(X_B|Y_A)P(Y_A) + P(X_B|Y_B)P(Y_B)}$$

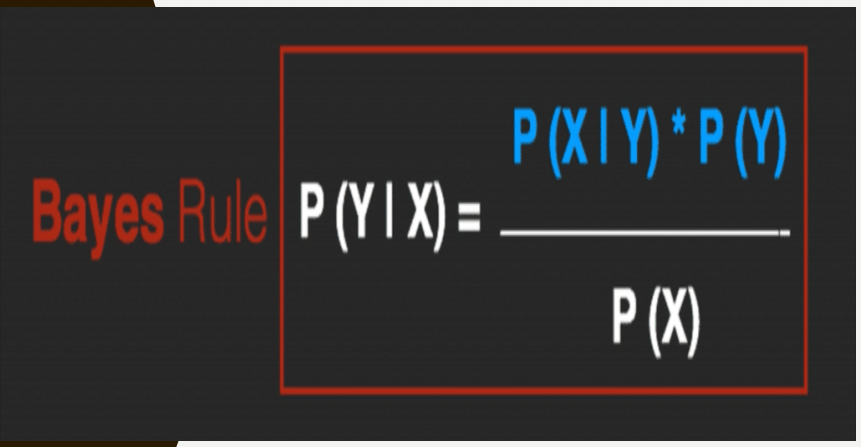
Probability that team B hosted the match won by team A is $P(X_B|Y_A) = 0.35$.

$$= \frac{0.35 \times 0.65}{(0.35 \times 0.65 + 0.75 \times 0.35)}$$

$$= 0.4642$$

The Naive Bayes

- The Bayes Rule provides the formula for the probability of Y given X. But, in real-world problems, you typically have multiple X variables.
- When the features are independent, we can extend the Bayes Rule to what is called **Naive Bayes**.
- It is called **'Naive'** because of the naive assumption that the X's are independent of each other. Regardless of its name, it's a powerful formula.



$$P(Y=k \mid X1..Xn) = \frac{P(X1 \mid Y=k) * P(X2 \mid Y=k) \ldots * P(Xn \mid Y=k) * P(Y=k)}{P(X1) * P(X2) \ldots * P(Xn)}$$

can be understood as ..

$$\text{Probability of Outcome I Evidence (Posterior Probability)} = \frac{\text{Probability of Likelihood of evidence} * \text{Prior}}{\text{Probability of Evidence}}$$

Probability of Evidence is same for all classes of Y

Q. Consider the given Datatset, Apply Navie-Bayes theorem and predict that if a fruit has the following properties then which type of fruit it is?

Fruit={Yellow, Sweet, Long}

Type	Long	Sweet	Yellow	Total
Banana	400	350	450	500
Orange	0	150	300	300
Other	100	150	50	200
Total	500	650	800	1000

Solution:

➢The objective of the classifier is to predict if a given fruit is a 'Banana' or 'Orange' or 'Other' when only the 3 features (long, sweet and yellow) are known.

Type	Long	Sweet	Yellow	Total
Banana	400	350	450	500
Orange	0	150	300	300
Other	100	150	50	200
Total	500	650	800	1000

- **<u>Step 1: Compute the 'Prior' probabilities for each of the class of fruits.</u>**
- For this case, let's compute from the training data. Out of 1000 records in training data, you have 500 Bananas, 300 Oranges and 200 Others. So the respective priors are 0.5, 0.3 and 0.2.
- **P(Y=Banana) = 500 / 1000 = 0.50**
- **P(Y=Orange) = 300 / 1000 = 0.30**
- **P(Y=Other) = 200 / 1000 = 0.20**

$$P(Y=k \mid X1..Xn) = \frac{P(X1 \mid Y=k) * P(X2 \mid Y=k) \ldots * P(Xn \mid Y=k) * P(Y=k)}{P(X1) * P(X2) \ldots * P(Xn)}$$

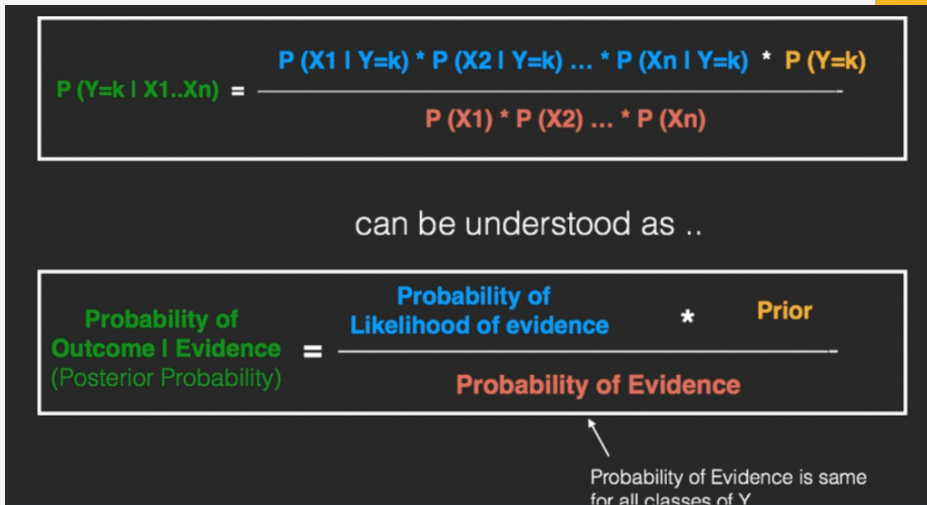
can be understood as ..

$$\text{Probability of Outcome I Evidence (Posterior Probability)} = \frac{\text{Probability of Likelihood of evidence} * \text{Prior}}{\text{Probability of Evidence}}$$

Probability of Evidence is same for all classes of Y

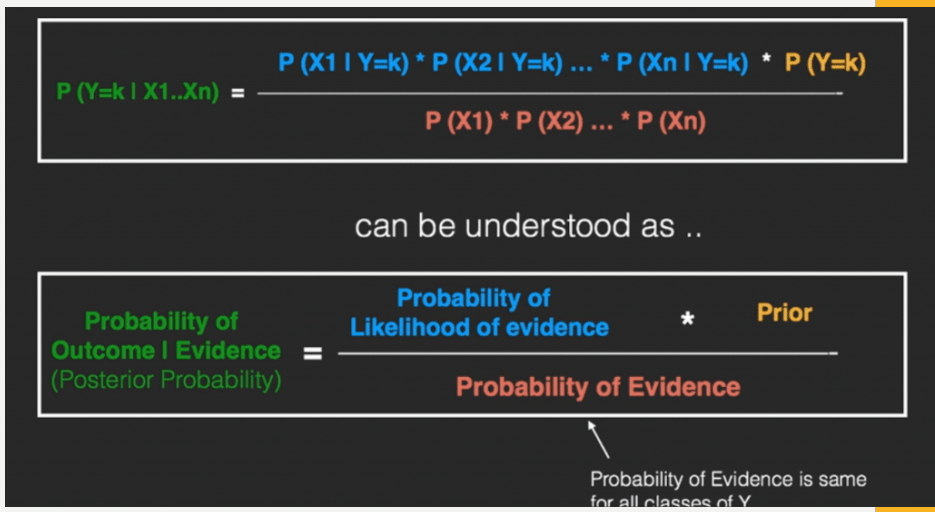
Type	Long	Sweet	Yellow	Total
Banana	400	350	450	500
Orange	0	150	300	300
Other	100	150	50	200
Total	500	650	800	1000

- **Step 2: Compute the probability of evidence that goes in the denominator.(OPTIONAL)**
- This is nothing but the product of P of Xs for all X.
- This is an **optional step** because the denominator is the same for all the classes and so will not affect the probabilities.
- **P(x1=Long) = 500 / 1000 = 0.50**
- **P(x2=Sweet) = 650 / 1000 = 0.65**
- **P(x3=Yellow) = 800 / 1000 = 0.80**



Type	Long	Sweet	Yellow	Total
Banana	400	350	450	500
Orange	0	150	300	300
Other	100	150	50	200
Total	500	650	800	1000

- **<u>Step 3: Compute the probability of likelihood of evidences that goes in the numerator.</u>**
- It is the product of conditional probabilities of the 3 features. If you refer back to the formula, it says P(X1 |Y=k). Here X1 is 'Long' and k is 'Banana'. That means the probability the fruit is 'Long' given that it is a Banana. In the above table, you have 500 Bananas. Out of that 400 is long. So, P(Long | Banana) = 400/500 = 0.8.
- Here, I have done it for Banana alone.
- **Probability of Likelihood for Banana**
  - **P(x1=Long | Y=Banana)** = 400 / 500 = 0.80
  - **P(x2=Sweet | Y=Banana)** = 350 / 500 = 0.70
  - **P(x3=Yellow | Y=Banana)** = 450 / 500 = 0.90
- So, the overall probability of Likelihood of evidence for Banana = **0.8 * 0.7 * 0.9 = 0.504**



- ***Similarly, you can compute the probabilities for 'Orange' and 'Other fruit'.***
  - **P(Orange |Long, Sweet and Yellow)=0**
  - **P(Other Fruit |Long, Sweet and Yellow)=0.01875**
- ***The denominator is the same for all 3 cases, so it's optional to compute.***

Step 4: If a fruit is *'Long', 'Sweet' and 'Yellow'*, what fruit is it?

$$P(\text{Banana} \mid \textit{Long, Sweet and Yellow}) = \frac{P(\text{Long} \mid \text{Banana}) * P(\text{Sweet} \mid \text{Banana}) * P(\text{Yellow} \mid \text{Banana}) \times P(\text{banana})}{P(\text{Long}) * P(\text{Sweet}) * P(\text{Yellow})}$$

$$= \frac{0.8 * 0.7 * 0.9 * 0.5}{P(\text{Evidence})} = 0.252/P(\text{Evidence})$$

P(Orange | *Long, Sweet and Yellow*) = 0, because P(Long | Orange) = 0

P(Other Fruit | *Long, Sweet and Yellow*) = 0.01875 / P(Evidence)

Answer: Banana - Since it has highest probability amongst the 3 classes

Q 5. Write A Program To Implement The Naïve Bayesian Classifier For A Sample Training Data Set Stored As A .CSV File. Compute The Accuracy Of The Classifier, Considering Few Test Data Sets.

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
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	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	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 classifiers

Naive Bayes is a statistical classification technique based on Bayes Theorem.

- It is one of the simplest supervised learning algorithms.
- Naive Bayes classifier is the fast, accurate and reliable algorithm.
- Naïve Bayes classifiers have high accuracy and speed on large datasets.

The Naive Bayes algorithm is called "**naive**" because it makes the assumption that the occurrence of a certain feature is independent of the occurrence of other features.

Naive Bayes classifier:

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

- Example: Play Tennis

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	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

The learning phase for tennis example

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	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

$P(\text{Play}=Yes) = 9/14$

$P(\text{Play}=No) = 5/14$

We have four variables, we calculate for each

Outlook	Play=*Yes*	Play=*No*
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Play=*Yes*	Play=*No*
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=*Yes*	Play=*No*
High	3/9	4/5
Normal	6/9	1/5

Wind	Play=*Yes*	Play=*No*
Strong	3/9	3/5
Weak	6/9	2/5

The ***test phase*** for the tennis example

- Test Phase
  - Given a new instance of variable values,
    x'=(Outlook=*Sunny*, Temperature=*Cool*, Humidity=*High*, Wind=*Strong*)
  - Given calculated Look up tables

P(Outlook=*Sunny*|Play=**Yes**) = 2/9
P(Temperature=*Cool*|Play=*Yes*) = 3/9
P(Huminity=*High*|Play=*Yes*) = 3/9
P(Wind=*Strong*|Play=*Yes*) = 3/9
P(Play=*Yes*) = 9/14

P(Outlook=*Sunny*|Play=**No**) = 3/5
P(Temperature=*Cool*|Play=*No*) = 1/5
P(Huminity=*High*|Play=*No*) = 4/5
P(Wind=*Strong*|Play=*No*) = 3/5
P(Play=*No*) = 5/14

  - **Use the MAP rule to calculate Yes or No**

P(*Yes*|**x'**): [P(*Sunny*|*Yes*)P(*Cool*|*Yes*)P(*High*|*Yes*)P(*Strong*|*Yes*)]P(Play=*Yes*) = 0.0053
P(*No*|**x'**): [P(*Sunny*|*No*) P(*Cool*|*No*)P(*High*|*No*)P(*Strong*|*No*)]P(Play=*No*) = 0.0206

Given the fact P(*Yes*|x') < P(*No*|x'), we label x' to be "*No*".

Algorithm

Algorithm:
NaiveBaiseClassifier(training_examples, New_Instance)

Each instance ***X*** is described by a conjunction of attribute values(a_i) and the target V can take j finite set of values.

a. For each value j in target estimate the P(Vj)
b. For each attribute in the training example estimate Estimate the P(ai|Vj)
c. Classify each instance as per the rule in equation

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

Where V_{NB} denotes the target value output by the naive Bayes classifier

d. Output V_{NB}

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	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

```python
import numpy as np
import math
import csv
```

```python
def read_data(filename):
    with open(filename,"r") as csvfile:
        datareader=csv.reader(csvfile)
        traindata=list(datareader)
    metadata=traindata[0]  #attributes name
    traindata=traindata[1:]  #training examples
    return (metadata, traindata)
```

```python
def splitdataset(dataset, splitratio):
    trainsize=int(len(dataset)*splitratio)
    trainset=[]
    testset=list(dataset)
    i=0
    while len(trainset)<trainsize:
        trainset.append(testset.pop(i))
    return [trainset,testset]

```

```
The attribute names of training data are: ['Outlook', 'Temperature', 'Humidity', 'Wind', 'Answer']
```

```
 The Training data set are:
['sunny' 'hot' 'high' 'weak' 'no']
['sunny' 'hot' 'high' 'strong' 'no']
['overcast' 'hot' 'high' 'weak' 'yes']
['rain' 'mild' 'high' 'weak' 'yes']
['rain' 'cool' 'normal' 'weak' 'yes']
['rain' 'cool' 'normal' 'strong' 'no']
['overcast' 'cool' 'normal' 'strong' 'yes']
['sunny' 'mild' 'high' 'weak' 'no']
['sunny' 'cool' 'normal' 'weak' 'yes']

 The Test data set are:
['rain' 'mild' 'normal' 'weak' 'yes']
['sunny' 'mild' 'normal' 'strong' 'yes']
['overcast' 'mild' 'high' 'strong' 'yes']
['overcast' 'hot' 'normal' 'weak' 'yes']
['rain' 'mild' 'high' 'strong' 'no']
```

```python
def classify(data,test):
    totalsize=data.shape[0]
    print("\n")
    print("training data size=",totalsize)
    print("test data size=",test.shape[0])

    countyes=0
    countno=0
    probyes=0
    probno=0
    print("\n")
    print("target \t count \t probolaity")
    for x in range(data.shape[0]):
        if data[x,data.shape[1]-1] =='yes':
            countyes+=1
        if data[x,data.shape[1]-1] =='no':
            countno+=1
    probYes=countyes/totalsize
    probNo=countno/totalsize

    print("Yes \t", countyes,"\t", probYes)
    print("No \t", countno,"\t", probNo)

    prob0=np.zeros((test.shape[1]-1))
    prob1=np.zeros((test.shape[1]-1))
```

```
training data size= 9
test data size= 5
```

target	count	probolaity
Yes	5	0.555555555555556
No	4	0.444444444444444

```
    accuracy=0
        print("\n")
    print("instance \t prediction \t target")

    for t in range(test.shape[0]):
        for k in range (test.shape[1]-1):
            count1=count0=0
            for j in range(data.shape[0]):
                #how many times appeared with no
                if test[t,k]==data[j,k] and data[j,data.shape[1]-1]=='no':
                    count0+=1
                #how many times appeared with yes
                if test[t,k]==data[j,k] and data[j,data.shape[1]-1]=='yes':
                    count1+=1
            prob0[k]=count0/countno
            prob1[k]=count1/countyes
        probno=probNo
        probyes=probYes
        for i in range(test.shape[1]-1):
            probno=probno*prob0[i]
            probyes=probyes*prob1[i]
        if probno>probyes:
            predict='no'
        else:
            predict='yes'

        print("  ",t+1," \t\t",predict," \t\t",test[t,test.shape[1]-1])
        if predict==test[t,test.shape[1]-1]:
            accuracy+=1
    finalaccuracy=(accuracy/test.shape[0])*100
    print("Accuracy=",finalaccuracy,"%")
```

```
 The Test data set are:
['rain' 'mild' 'normal' 'weak' 'yes']
['sunny' 'mild' 'normal' 'strong' 'yes']
['overcast' 'mild' 'high' 'strong' 'yes']
['overcast' 'hot' 'normal' 'weak' 'yes']
['rain' 'mild' 'high' 'strong' 'no']
```

$$v_{NB}(no) = P(no)\ P(sunny|no)\ P(cool|no)\ P(high|no)\ P(strong|no)$$

$$v_{NB}(yes) = P(yes)\ P(sunny|yes)\ P(cool|yes)\ P(high|yes)\ P(strong|yes)$$

instance	prediction	target
1	yes	yes
2	no	yes
3	yes	yes
4	yes	yes
5	no	no

Accuracy= 80.0 %

```
metadata,traindata=read_data("D://id3.csv")
print("\n The attribute names of training data are:",metadata)
splitratio=0.7
trainset, testset=splitdataset(traindata, splitratio)
training=np.array(trainset)
print("\n The Training data set are:")
for x in training:
    print(x)

testing=np.array(testset)
print("\n The Test data set are:")
for x in testing:
    print(x)

classify(training, testing)
```

```
The attribute names of training data are: ['Outlook', 'Temperature', 'Humidity', 'Wind', 'Answer']

 The Training data set are:
['sunny' 'hot' 'high' 'weak' 'no']
['sunny' 'hot' 'high' 'strong' 'no']
['overcast' 'hot' 'high' 'weak' 'yes']
['rain' 'mild' 'high' 'weak' 'yes']
['rain' 'cool' 'normal' 'weak' 'yes']
['rain' 'cool' 'normal' 'strong' 'no']
['overcast' 'cool' 'normal' 'strong' 'yes']
['sunny' 'mild' 'high' 'weak' 'no']
['sunny' 'cool' 'normal' 'weak' 'yes']

 The Test data set are:
['rain' 'mild' 'normal' 'weak' 'yes']
['sunny' 'mild' 'normal' 'strong' 'yes']
['overcast' 'mild' 'high' 'strong' 'yes']
['overcast' 'hot' 'normal' 'weak' 'yes']
['rain' 'mild' 'high' 'strong' 'no']
```

```
training data size= 9
test data size= 5

target   count   probolaity
Yes      5       0.5555555555555556
No       4       0.4444444444444444

instance         prediction      target
   1             yes             yes
   2             no              yes
   3             yes             yes
   4             yes             yes
   5             no              no
Accuracy= 80.0 %
```

Naïve Bayes Classifier

- Naive Bayes is a statistical classification technique based on Bayes Theorem.
- It is one of the simplest supervised learning algorithms.
- Naive Bayes classifier is the fast, accurate and reliable algorithm.
- Naive Bayes classifier calculates the probability of an event in the following steps:

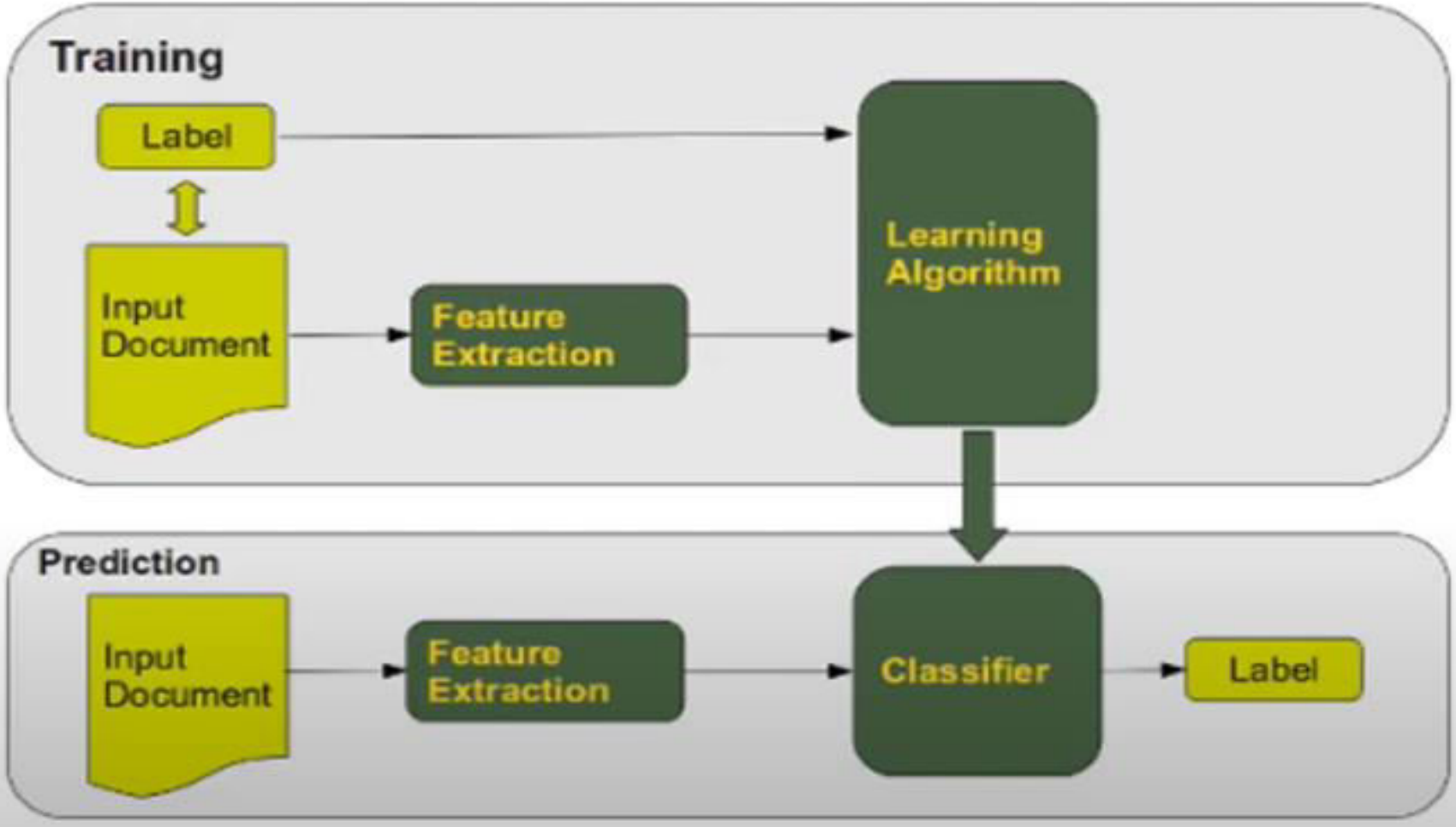
Step 1: Calculate the prior probability for given class labels

Step 2: Find Likelihood probability with each attribute for each class

Step 3: Put these value in Bayes Formula and calculate posterior probability.

Step 4: See which class has a higher probability, given the input belongs to the higher probability class.

What is text classification ?



Source: https://www.python-course.eu/text_classification_introduction.php

Evaluation Metrics

These metrics are used to evaluate the results of classifications:

- Accuracy
- Precision
- Recall

Evaluation Metrics

Accuracy

Accuracy is a statistical measure which is defined as the quotient of correct predictions (both True positives (TP) and True negatives (TN)) made by a classifier divided by the sum of all predictions made by the classifier, including False positives (FP) and False negatives (FN).

The formula:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Confusion Matrix

A confusion matrix, also called a contingency table or error matrix, is used to visualize the performance of a classifier.

Confusion Matrix		Predicted classes: negative	Predicted classes: positive
Actual classes	negative	TN	FP
Actual classes	positive	FN	TP

Evaluation Metrics

- **Precision:** Precision is the ratio of the correctly identified positive cases to all the predicted positive cases, i.e. the correctly and the incorrectly predicted cases as positive. Precision is the fraction of retrieved documents that are relevant to the query.
- The formula:

$$precision = \frac{TP}{TP + FP}$$

Confusion Matrix		Predicted classes: negative	Predicted classes: positive
Actual classes	negative	TN	FP
	positive	FN	TP

- **Recall**, also known as sensitivity, is the ratio of the correctly identified positive cases to all the actual positive cases, which is the sum of the "False Negatives" and "True Positives".
- The formula:

$$recall = \frac{TP}{TP + FN}$$

Naïve Bayes classifier algorithm for text classification

LEARN_NAIVE_BAYES_TEXT(*Examples*, *V*)

Examples is a set of text documents along with their target values. V is the set of all possible target values. This function learns the probability terms $P(w_k|v_j)$, describing the probability that a randomly drawn word from a document in class v_j will be the English word w_k. It also learns the class prior probabilities $P(v_j)$.

1. collect all words, punctuation, and other tokens that occur in Examples

- *Vocabulary* ← the set of all distinct words and other tokens occurring in any text document from *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
  - $docs_j$ ← the subset of documents from *Examples* for which the target value is v_j
  - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
  - $Text_j$ ← a single document created by concatenating all members of $docs_j$
  - n ← total number of distinct word positions in $Text_j$
  - for each word w_k in *Vocabulary*
    - n_k ← number of times word w_k occurs in $Text_j$
    - $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

CLASSIFY_NAIVE_BAYES_TEXT(*Doc*)

Return the estimated target value for the document Doc. a_i denotes the word found in the ith position within Doc.

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

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

Example : Movie Review

Training Data

Examples	Text	Class
1	I loved the movie	+
2	I hated the movie	-
3	a great movie. good movie	+
4	poor acting	-
5	a great acting. good movie	+

Vocabulary

< I, loved, the, movie, hated, a, great, good, poor, acting >

Test Data

I hated the poor acting

Example : Text Classification

$docs_j \leftarrow$ the subset of documents from *Examples* for which the target value is v_j

Documents with positive (+) class

Docs	I	loved	the	movie	hated	a	great	good	poor	acting	Class
1	1	1	1	1							+
3				2		1	1	1			+
5				1		1	1	1		1	+

$P(v_j) \leftarrow |docs_j| / |\text{Examples}|$ $\qquad P(+) = \frac{3}{5} = 0.6$

Example : Text Classification

$Text_j \leftarrow$ a single document created by concatenating all members of $docs_j$

$Text_j \leftarrow$ < I loved the movie a great movie. good movie great acting. good movie >

Docs	I	loved	the	movie	hated	a	great	good	poor	acting	Class
1	1	1	1	1							+
3				2		1	1	1			+
5				1		1	1	1		1	+

$n \leftarrow$ total number of word positions in $Text_j$

$n \leftarrow 14$

Example 1: Text Classification

- for each word w_k in *Vocabulary*

$n_k \leftarrow$ number of times word $\boldsymbol{w_k}$ occurs in $Text_j$ $\qquad P(w_k|v_j) \leftarrow (n_k + 1) / (n + |Vocabulary|)$

Docs	I	loved	the	movie	hated	a	great	good	poor	acting	Class
1	1	1	1	1							+
3				2		1	1	1			+
5				1		1	1	1		1	+

$$P(I|+) = \frac{1+1}{14+10} = 0.0833 \qquad P(a|+) = \frac{2+1}{14+10} = 0.125$$

$$P(loved|+) = \frac{1+1}{14+10} = 0.0833 \qquad P(great|+) = \frac{2+1}{14+10} = 0.125$$

$$P(the|+) = \frac{1+1}{14+10} = 0.0833 \qquad P(good|+) = \frac{2+1}{14+10} = 0.125$$

$$P(movie|+) = \frac{4+1}{14+10} = 0.2083 \qquad P(poor|+) = \frac{0+1}{14+10} = 0.0416$$

$$P(hated|+) = \frac{0+1}{14+10} = 0.0416 \qquad P(acting|+) = \frac{1+1}{14+10} = 0.0833$$

Example 1: Text Classification

$docs_j \leftarrow$ the subset of documents from *Examples* for which the target value is v_j

Documents with positive (-) class

docs	I	loved	the	movie	hated	a	great	good	poor	acting	Class
2	1		1	1	1						-
4									1	1	-

$$P(v_j) \leftarrow |docs_j| / |\text{Examples}| \qquad P(-) = \frac{2}{5} = 0.4$$

Example 1: Text Classification

$Text_j \leftarrow$ a single document created by concatenating all members of $docs_j$

$Text_j \leftarrow$ < I hated the movie poor acting >

docs	I	loved	the	movie	hated	a	great	good	poor	acting	Class
2	1		1	1	1						-
4									1	1	-

$n \leftarrow$ total number of distinct word positions in $Text_j$

$n \leftarrow 6$

Example 1: Text Classification

- for each word w_k in *Vocabulary*

$n_k \leftarrow$ number of times word $\boldsymbol{w_k}$ occurs in $Text_j$ $\qquad P(w_k|v_j) \leftarrow (n_k + 1) / (n + |Vocabulary|)$

docs	I	loved	the	movie	hated	a	great	good	poor	acting	Class
2	1		1	1	1						-
4									1	1	-

$$P(I|-) = \frac{1+1}{6+10} = 0.125$$

$$P(loved|-) = \frac{0+1}{6+10} = 0.0625$$

$$P(the|-) = \frac{1+1}{6+10} = 0.125$$

$$P(movie|-) = \frac{1+1}{6+10} = 0.125$$

$$P(hated|-) = \frac{1+1}{6+10} = 0.125$$

$$P(a|-) = \frac{0+1}{6+10} = 0.0625$$

$$P(great|-) = \frac{0+1}{6+10} = 0.0625$$

$$P(good|-) = \frac{0+1}{6+10} = 0.0625$$

$$P(poor|-) = \frac{1+1}{6+10} = 0.125$$

$$P(acting|-) = \frac{1+1}{6+10} = 0.125$$

Example : Text Classification

Let's classify the new document

I hated the poor acting

If $V_j = +$

then,

$= P(+)\, P(I \mid +)\, P(\text{hated} \mid +)\, P(\text{the} \mid +)\, P(\text{poor} \mid +)\, P(\text{acting} \mid +)$

$= 0.6 * 0.0833 * 0.0416 * 0.0833 * 0.0416 * 0.0833$

$= 6.03 \times 10^{-7}$

If $V_j = -$

then,

$= P(-)\, P(I \mid -)\, P(\text{hated} \mid -)\, P(\text{the} \mid -)\, P(\text{poor} \mid -)\, P(\text{acting} \mid -)$

$= 0.4 * 0.125 * 0.125 * 0.125 * 0.125 * 0.125$

$= 1.22 \times 10^{-5}$

$1.22 \times 10^{-5} > 6.03 \times 10^{-7}$

So, the new document belongs to (–) class

CSV: data6.csv

1	I love this sandwich	pos
2	This is an amazing place	pos
3	I feel very good about these beers	pos
4	This is my best work	pos
5	What an awesome view	pos
6	I do not like this restaurant	neg
7	I am tired of this stuff	neg
8	I can't deal with this	neg
9	He is my sworn enemy	neg
10	My boss is horrible	neg
11	This is an awesome place	pos
12	I do not like the taste of this juice	neg
13	I love to dance	pos
14	I am sick and tired of this place	neg
15	What a great holiday	pos
16	That is a bad locality to stay	neg
17	We will have good fun tomorrow	pos
18	I went to my enemy's house today	neg

Program:

```
1 import pandas as pd
2 msg=pd.read_csv('D:\\data6.csv',names=['message','label']) #Tabular form data
3 print('Total instances in the dataset:',msg.shape[0])
4
5 msg['labelnum']=msg.label.map({'pos':1,'neg':0})
6 X=msg.message
7 Y=msg.labelnum
```

```
Total instances in the dataset: 18
```

	message	label	labelnum
0	I love this sandwich	pos	1
1	This is an amazing place	pos	1
2	I feel very good about these beers	pos	1
3	This is my best work	pos	1
4	What an awesome view	pos	1
5	I do not like this restaurant	neg	0
6	I am tired of this stuff	neg	0
7	I can't deal with this	neg	0
8	He is my sworn enemy	neg	0
9	My boss is horrible	neg	0
10	This is an awesome place	pos	1
11	I do not like the taste of this juice	neg	0
12	I love to dance	pos	1
13	I am sick and tired of this place	neg	0
14	What a great holiday	pos	1
15	That is a bad locality to stay	neg	0
16	We will have good fun tomorrow	pos	1
17	I went to my enemy's house today	neg	0

Program:

```
13 # Splitting the dataset into train and test data
14 from sklearn.model_selection import train_test_split
15 xtrain,xtest,ytrain,ytest=train_test_split(X,Y)
16
17 print('\nDataset is split into Training and Testing samples')
18 print('Total training instances :', ytrain.shape[0])
19 print('Total testing instances :', ytest.shape[0])
20
21 # Output of count vectoriser is a sparse matrix
22 # CountVectorizer - stands for 'feature extraction'
23 from sklearn.feature_extraction.text import CountVectorizer
24 count_vect = CountVectorizer()
25 xtrain_dtm = count_vect.fit_transform(xtrain) #Sparse matrix
26 xtest_dtm = count_vect.transform(xtest)
27 print('\nTotal features extracted using CountVectorizer:',xtrain_dtm.shape[1])
28
29 print('\nThe words or Tokens in the tex documents\n')
30 print(count_vect.get_feature_names())
31
```


```
Total features extracted using CountVectorizer: 46

The words or Tokens in the tex documents

['about', 'am', 'amazing', 'an', 'and', 'awesome', 'beers', 'best', 'can', 'deal', 'do', 'enemy', 'feel', 'fun', 'good', 'grea
t', 'have', 'he', 'holiday', 'house', 'is', 'juice', 'like', 'my', 'not', 'of', 'place', 'restaurant', 'sick', 'sworn', 'tast
e', 'the', 'these', 'this', 'tired', 'to', 'today', 'tomorrow', 'very', 'view', 'we', 'went', 'what', 'will', 'with', 'work']
```

Program:

```
34 # Training Naive Bayes (NB) classifier on training data.
35 from sklearn.naive_bayes import MultinomialNB
36 clf = MultinomialNB().fit(xtrain_dtm,ytrain)
37 predicted = clf.predict(xtest_dtm)
38
39 #printing accuracy metrics
40 from sklearn import metrics
41 print('\nAccuracy metrics')
42 print('Accuracy of the classifer is',metrics.accuracy_score(ytest,predicted))
43
44 print('Recall :',metrics.recall_score(ytest,predicted), '\nPrecison :',metrics.precision_score(ytest,predicted))
45 print('Confusion matrix')
46 print(metrics.confusion_matrix(ytest,predicted))
```

```
Accuracy metrics
==================
Accuracy of the classifer is 0.6
Recall : 0.5
Precison : 1.0
Confusion matrix
==================
[[1 0]
 [2 2]]
```

BAYESIAN NETWORK

BAYESIAN BELIEF NETWORKS

- **motivation**
  - naive Bayes classifier makes significant use of the assumption of conditional independence
  - this assumption dramatically reduces the complexity of the learning task
  - however, in many cases this assumption is overly restrictive

- **Bayesian Belief Network**
  - describes probability distribution governing a set of variables by specifying a **set of conditional independence assumptions** along with a **set of conditional probabilites**
  - conditional independence assumption applies only to subsets of the variables

BAYESIAN BELIEF NETWORKS-NOTATION

- Bayesian Belief Networks describe the probability distribution over a set of variables
- arbitrary set of random variables $Y_1, ..., Y_n$ where $V(Y_i)$ is the set of possible values for Y_i
- *joint space*: $V(Y_1) \times V(Y_2) \times ... \times V(Y_n)$
- *joint probability distribution* specifies the probability for each of the possible variable bindings for the tuple $< Y_1, Y_2, ..., Y_n >$

CONDITIONAL INDEPENDENCE

- Let us understand the idea of 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; that is, if

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

where $x_i \in V(X)$, $y_j \in V(Y)$, and $z_k \in V(Z)$.

- We commonly write the above expression in abbreviated form as

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

CONDITIONAL INDEPENDENCE

- This definition of conditional independence can be extended to sets of variables as well.
- We say that the set of variables **X1 . . . Xi** is conditionally independent of the set of variables **Y1 . . . Ym** given the set of variables **Z1 . . . Zn** if

$$\mathbf{P(X1 \ldots Xi|Y1 \ldots Ym, z1 \ldots Z,) = P(X1 \ldots Xi|Z1 \ldots Zn)}$$

- There is a relation between this definition and our use of conditional independence in the definition of the naive Bayes classifier.
- The naive Bayes classifier assumes that the instance attribute **A1** is conditionally independent of instance attribute **A2** given the target value **V.**
- This allows the naive Bayes classifier to calculate **P(A1, A2|V)** in Equation

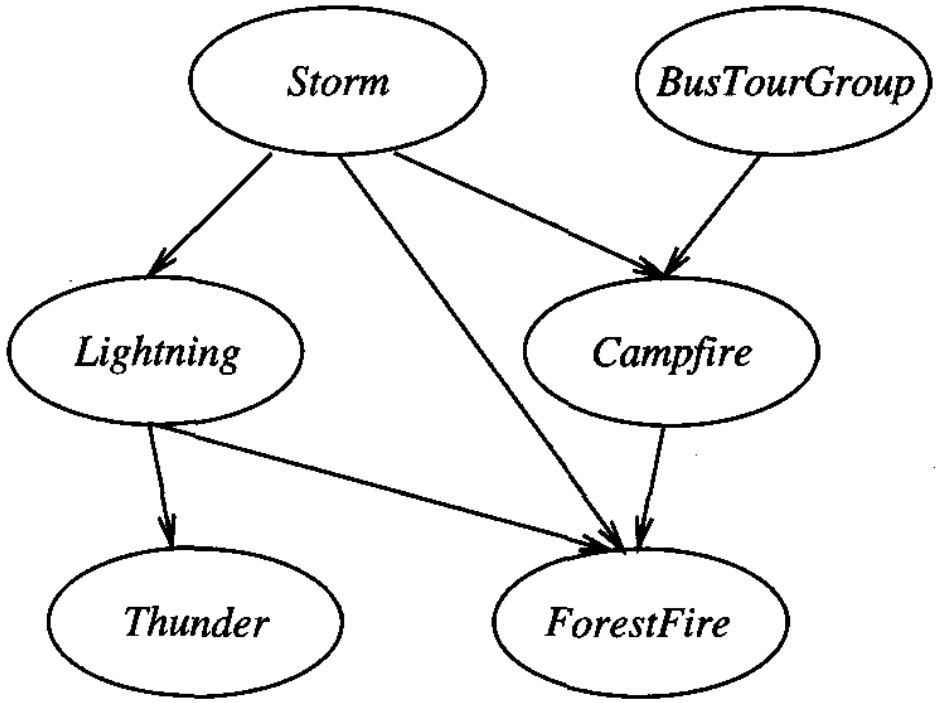
$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod P(a_i|v_j)$$

as follows:

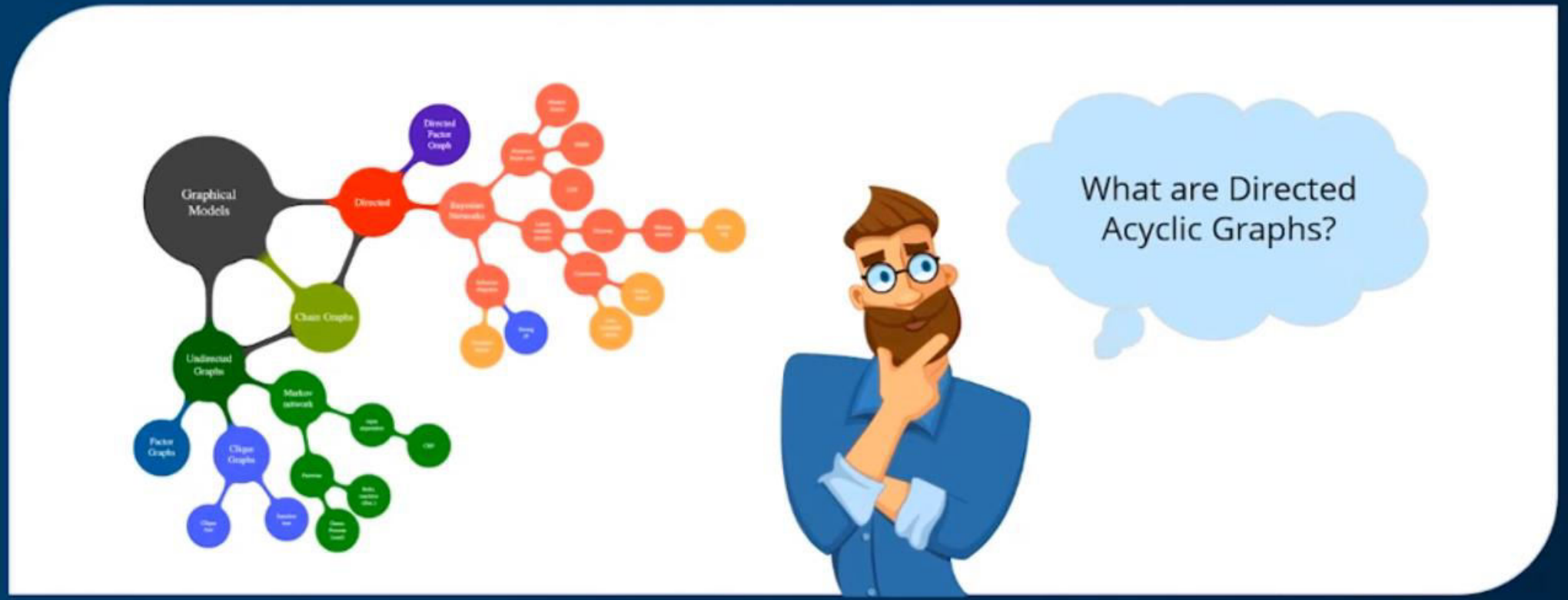
$$P(A_1, A_2|V) = P(A_1|A_2, V)P(A_2|V)$$
$$= P(A_1|V)P(A_2|V)$$

REPRESENTATION

- ***A Bayesian belief network*** (Bayesian network for short) represents the joint probability distribution for a set of variables.
- For example, the Bayesian network in Figure below represents the joint probability distribution over the boolean variables ***Storm, Lightning, Thunder, ForestFire, Campjre,*** and ***BusTourGroup.***

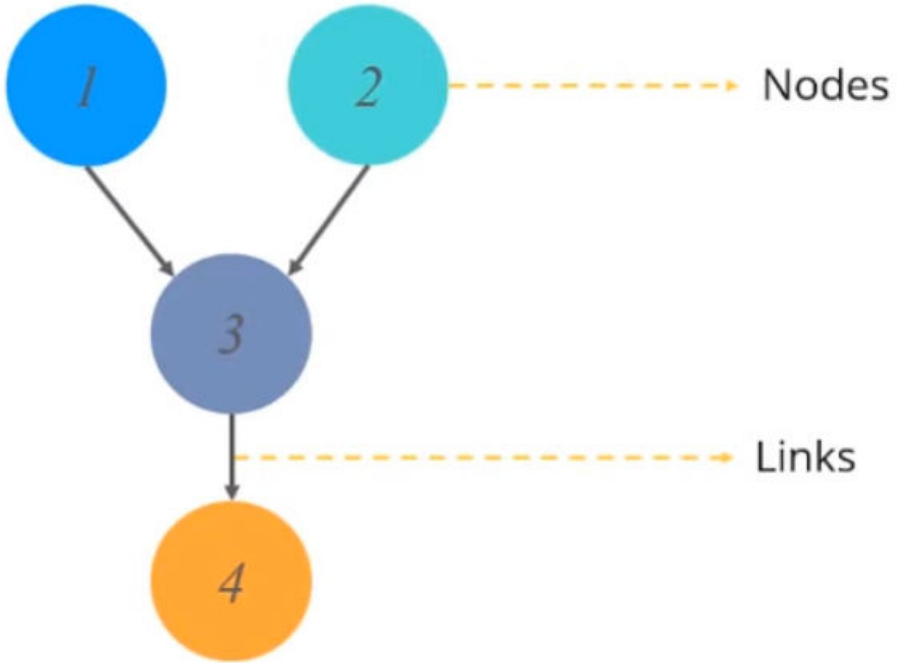


- A Bayesian network represents the joint probability distribution by specifying a set of conditional independence assumptions (represented by a directed acyclic graph), together with sets of local conditional probabilities
- Each variable in the joint space is represented by a node in the Bayesian network



What Is A Bayesian Network?

A Bayesian Network falls under the category of Probabilistic Graphical Modelling (PGM) technique that is used to compute uncertainties by using the concept of probability.



A DAG models the uncertainty of an event occurring based on the Conditional Probability Distribution (CDP) of each random variable

What Is A Directed Acyclic Graph?

A Directed Acyclic Graph is used to represent a Bayesian Network and like any other statistical graph, a DAG contains a set of nodes and links, where the links denote the relationship between the nodes.

REPRESENTATION

- For each variable two types of information are specified.
  - **First, the network arcs** represent the assertion that the variable is conditionally independent of its nondescendants in the network given its immediate predecessors in the network. (i.e., **Xj** is a ***descendant*** of ***Y*** if there is a directed path from ***Y*** to **X.**)
  - **Second, a conditional probability table** is given for each variable, describing the probability distribution for that variable given the values of its immediate predecessors.
- The joint probability for any desired assignment of values ***(Yl,*** . . . , ***Yn)*** to the tuple of network variables ***<YI*** . . . ***Yn>*** can be computed by the formula

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

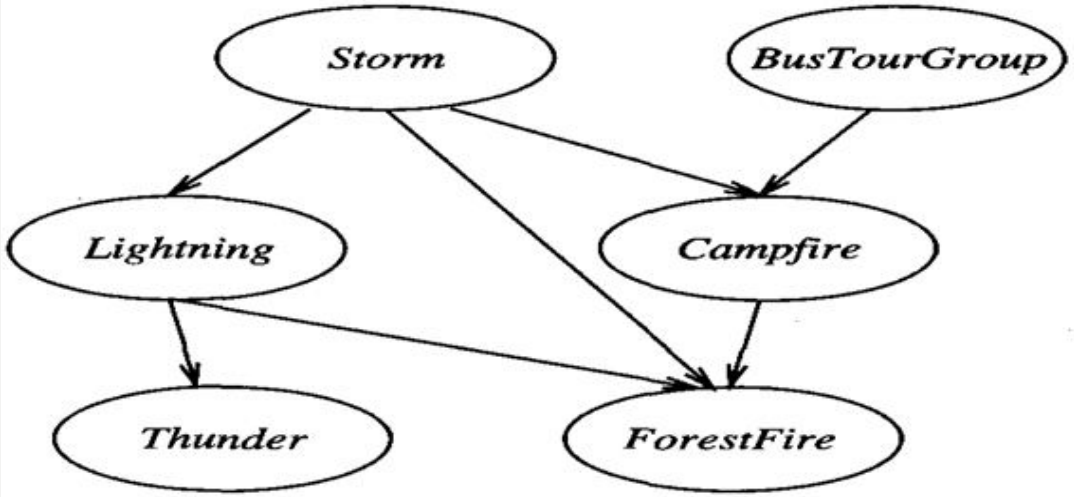
REPRESENTATION

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

- where **Parents(Yi)** denotes the set of immediate predecessors of ***Yi*** in the network.
- Note the values of **P(yi|Parents(Yi))** are precisely the values stored in the conditional probability table associated with node ***Yi.***

ILLUSTRATION

1.DAG



2. CONDITIONAL PROBABILITY TABLE

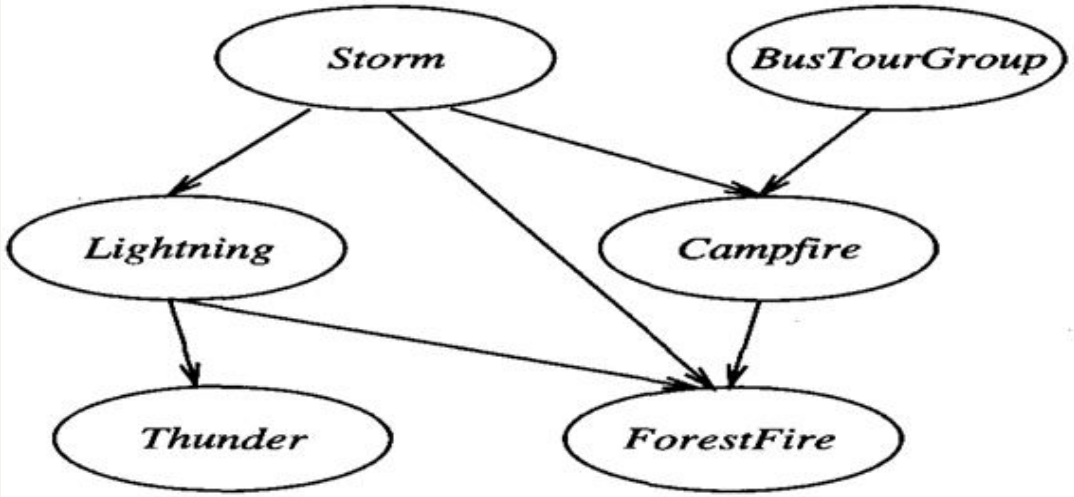
	S,B	$S,\neg B$	$\neg S,B$	$\neg S,\neg B$
C	0.4	0.1	0.8	0.2
$\neg C$	0.6	0.9	0.2	0.8

Campfire

- The Bayesian network in Figure represents the joint probability distribution over the boolean variables ***Storm, Lightning, Thunder, ForestFire, Campfire,*** and ***BusTourGroup.***
- Consider the node ***Campfire.***
- The network nodes and arcs represent the assertion that ***Campfire*** is conditionally independent of its nondescendants ***Lightning*** and ***Thunder,*** given its immediate parents ***Storm*** and ***BusTourGroup.***

ILLUSTRATION

1.DAG



2. CONDITIONAL PROBABILITY TABLE

	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

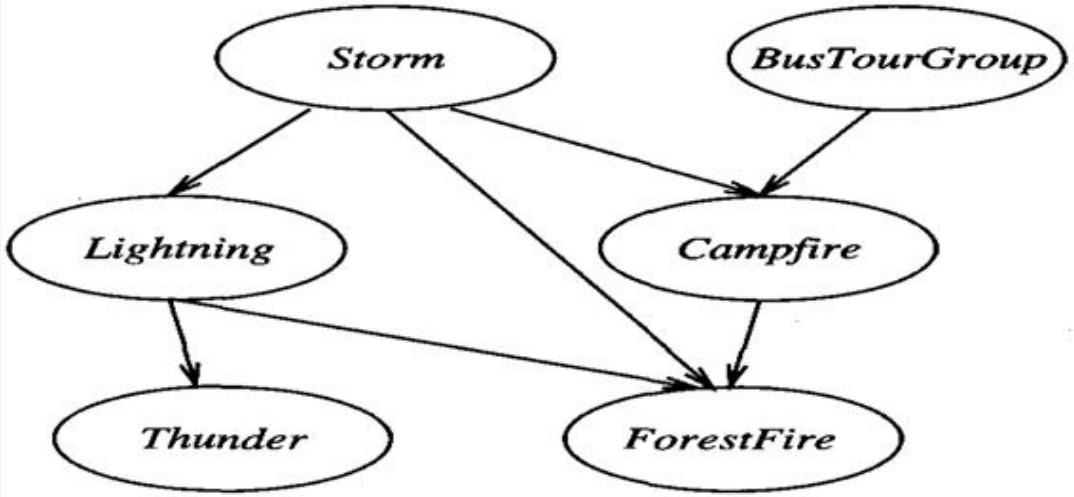
Campfire

- This means that once we know the value of the variables ***Storm*** and ***BusTourGroup,*** the variables ***Lightning*** and ***Thunder*** provide no additional information about ***Campfire.***
- The right side of the figure shows the conditional probability table associated with the variable ***Campfire***.
- The top left entry in this table, for example, expresses the assertion that

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

ILLUSTRATION

1.DAG



2. CONDITIONAL PROBABILITY TABLE

	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

Campfire

- Note this table provides only the conditional probabilities of ***Campfire*** given its parent variables ***Storm*** and ***BusTourGroup.***
- The set of local conditional probability tables for all the variables, together with the set of conditional independence assumptions described by the network, describe the full joint probability distribution for the network.

INFERENCE

1.Infer the value of some target variable given the observed values of other variables

- Pr(F|S,B,L,C,T)

2. Infer the probability distribution of the target variable

3. Infer subset of variables when some other variables are known

Inference Efficiency

- Exact Inference for arbitrary networks is NP-hard
- Approximate inference sacrifices precision to gain efficiency
  - Monte Carlo method for randomly sampling unobserved variables
  - Approximate methods can also be NP-hard but useful in many cases

LEARNING BAYESIAN BELIEF NETWORKS

- Learning Bayesian Networks from training data
- Several different settings for this problem

1 Network structure

- given in advance or
- inferred from training data

2 Network variables

- are observable in training data or
- might be unobservable

LEARNING BAYESIAN BELIEF NETWORKS

Network structure given in advance

1. **Variables are fully observable in training data**
    - Estimate conditional probability table entries as for a NB
2. **Only some variable values are observable**
    - More difficult
    - Similar to learning weights for hidden units in ANNs
        -ANNs have input and output values specified
        -Hidden unit values are learnt from training examples

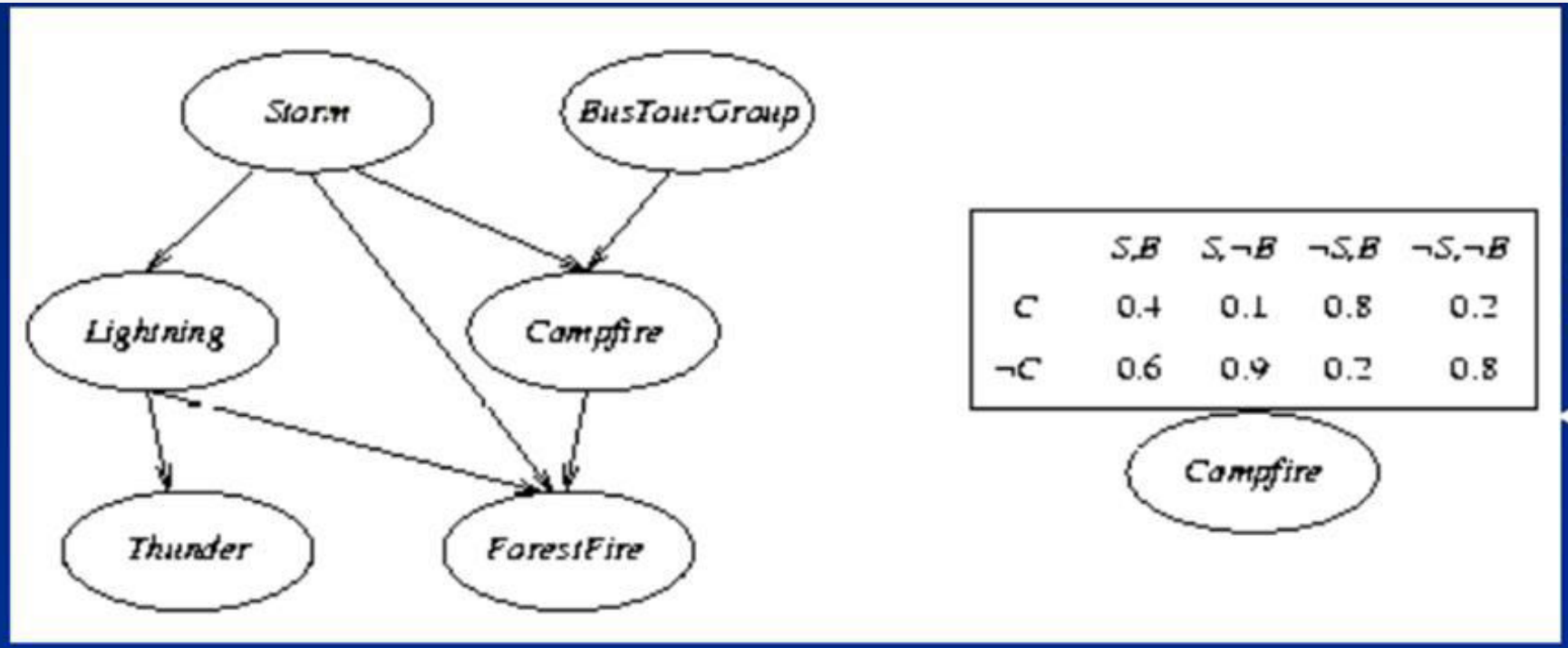
GRADIENT ASCENT FOR LEARNING PROBABILITIES

- Learns entries in the conditional probability tables
- Gradient Ascent searches through the space of all possible entries for the conditional probability tables
- Objective function maximized during ascent is the probability P(D|h) of the observed training data given the hypothesis h
- Corresponds to searching for the maximum likelihood hypothesis

GRADIENT ASCENT TRAINING OF BAYESIAN NETWORKS

- Network structure given but only some variables are observable
- Learns entries in conditional probability tables
- Searches through a space of hypotheses that corresponds to the set of all possible entries for the conditional probability tables
- Objective function is maximized: P(D|h) probability of observed data given the hypothesis h
- Maximize P(D|h) with respect to the parameters that define the conditional probability tables of the Bayesian network
- w_{ijk} is a single entry in one of the tables in the known structure of the Bayesian network
  - i = index of variable
  - j = index of value of variable
  - k = index of parent

BAYESIAN NETWORK GRADIENT ASCENT NOTATION



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

Campfire

Conditional Probability Table for variable Y_4

w_{ijk} is the probability that Y_i will take on value y_{ij} given that its immediate parents U_i take on the values given by u_{ik}

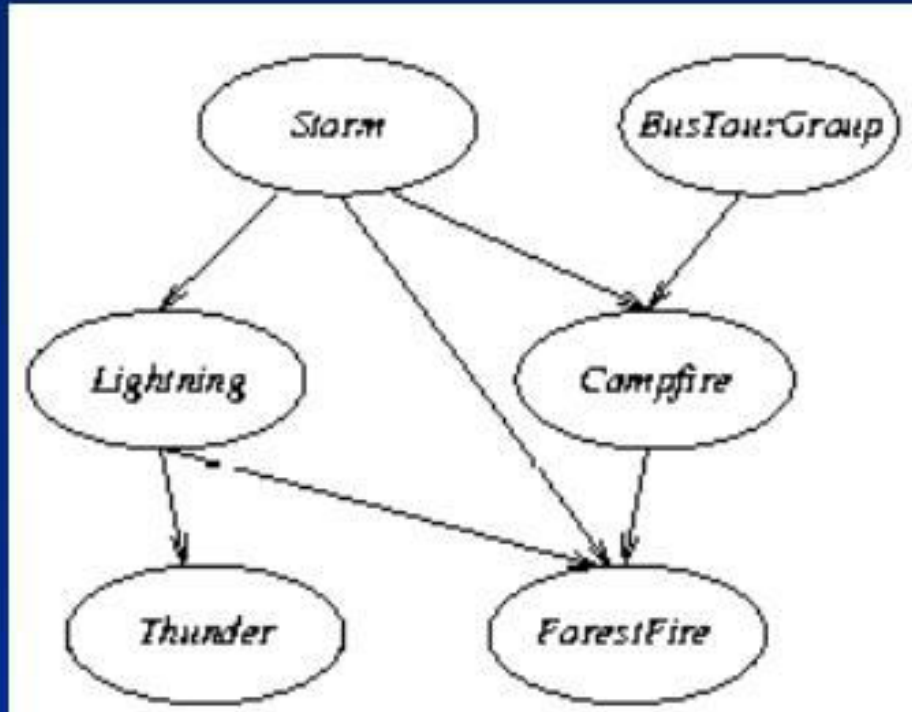
If w_{ijk} is top right entry of table, Y_i is the variable Campfire, U_i are the parents-tuple <Storm,BusTourGroup>, y_{ij}=True, and u_{ik}=<False, False>

Gradient Ascent Training of Bayesian Networks

- Maximize P(D|h) by following the gradient of ln P(D|h)
- w_{ijk} is a single entry in one of the tables in the Bayesian network
- It can be shown that that each of these derivatives can be calculated as

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

Gradient Ascent calculation example



Y_3

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

Conditional Probability Table for variable Y_4

To calculate derivative of ln p(D|h)

with respect to upper-rightmost entry in table

we have to calculate

P(Campfire=True,Storm=False,BusTourGroup=False|d) for each training example d in D

Gradient Ascent Training Procedure

- Two-step Procedure
  1. Update each w_{ijk} using training data D

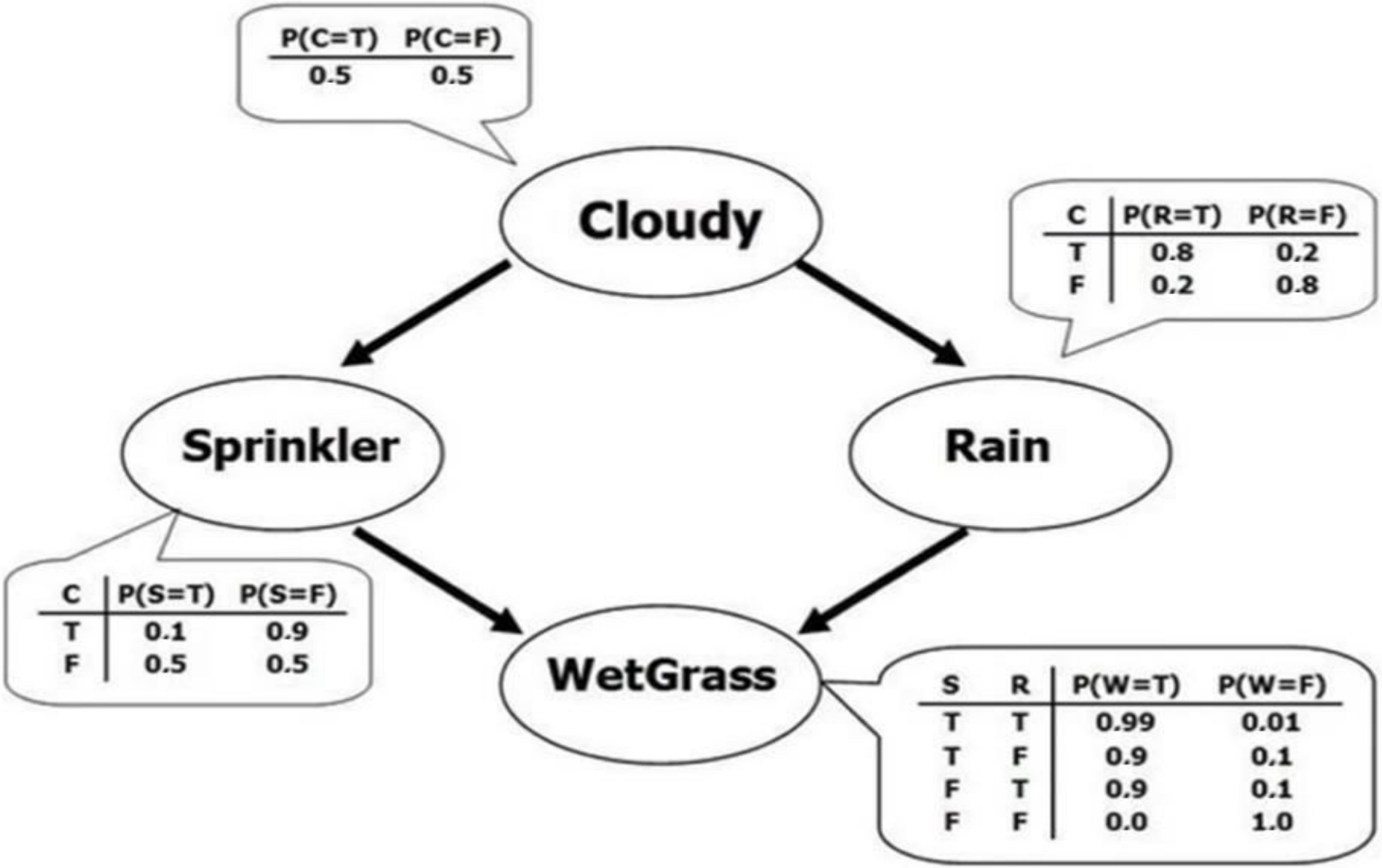
$$w_{ijk} \leftarrow w_{ijk} + \eta \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik} \mid d)}{w_{ijk}}$$

where η is the learning rate

  2. Renormalize weights w_{ijk} to assure

$$\Sigma_j w_{ijk} = 1 \qquad 0 \leq w_{ijk} \leq 1$$

A related example



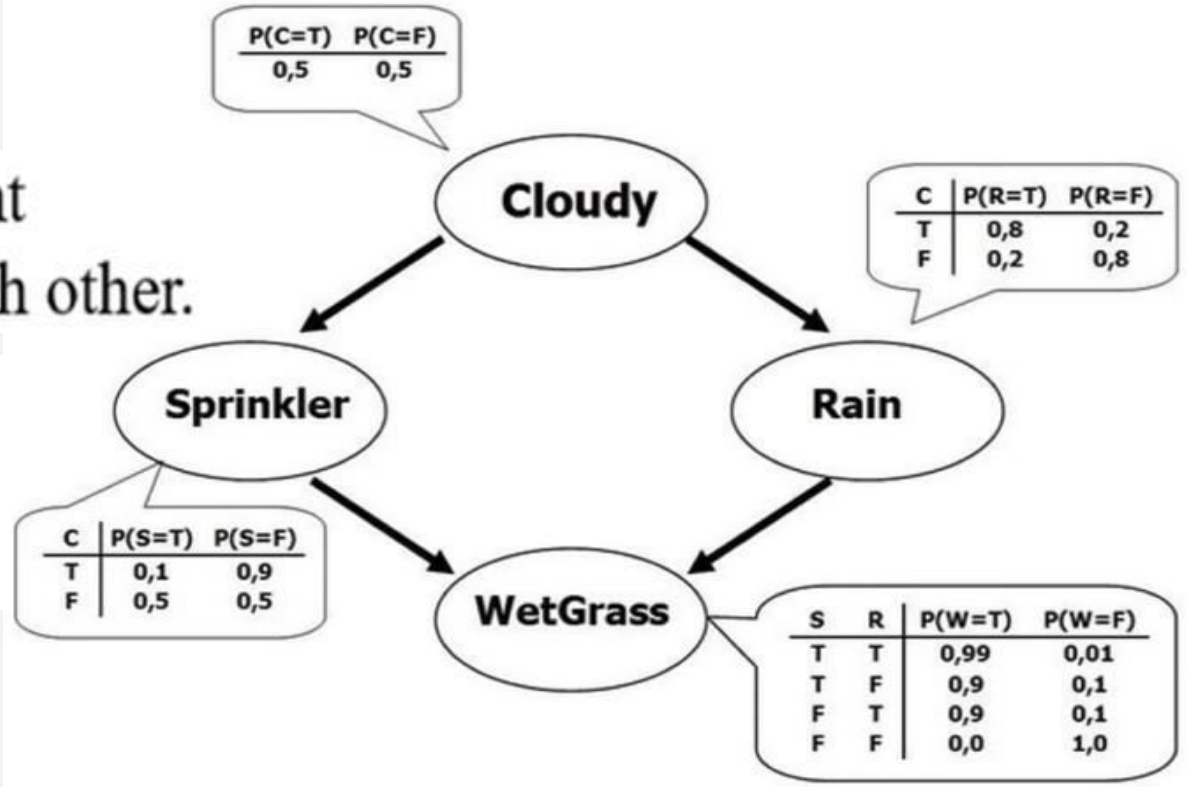
Conditional Independence

Nodes that are not connected by any path represent variables that are conditionally independent of each other.

e.g. Sprinkler and Rain

in the related example are conditionally independent

P(Sprinkler|Cloudy, Rain) = P(Sprinkler|Cloudy)



This joint probabilities for the network can be calculated as.

P(Wetgrass, Sprinkler, Cloudy, Rain)=

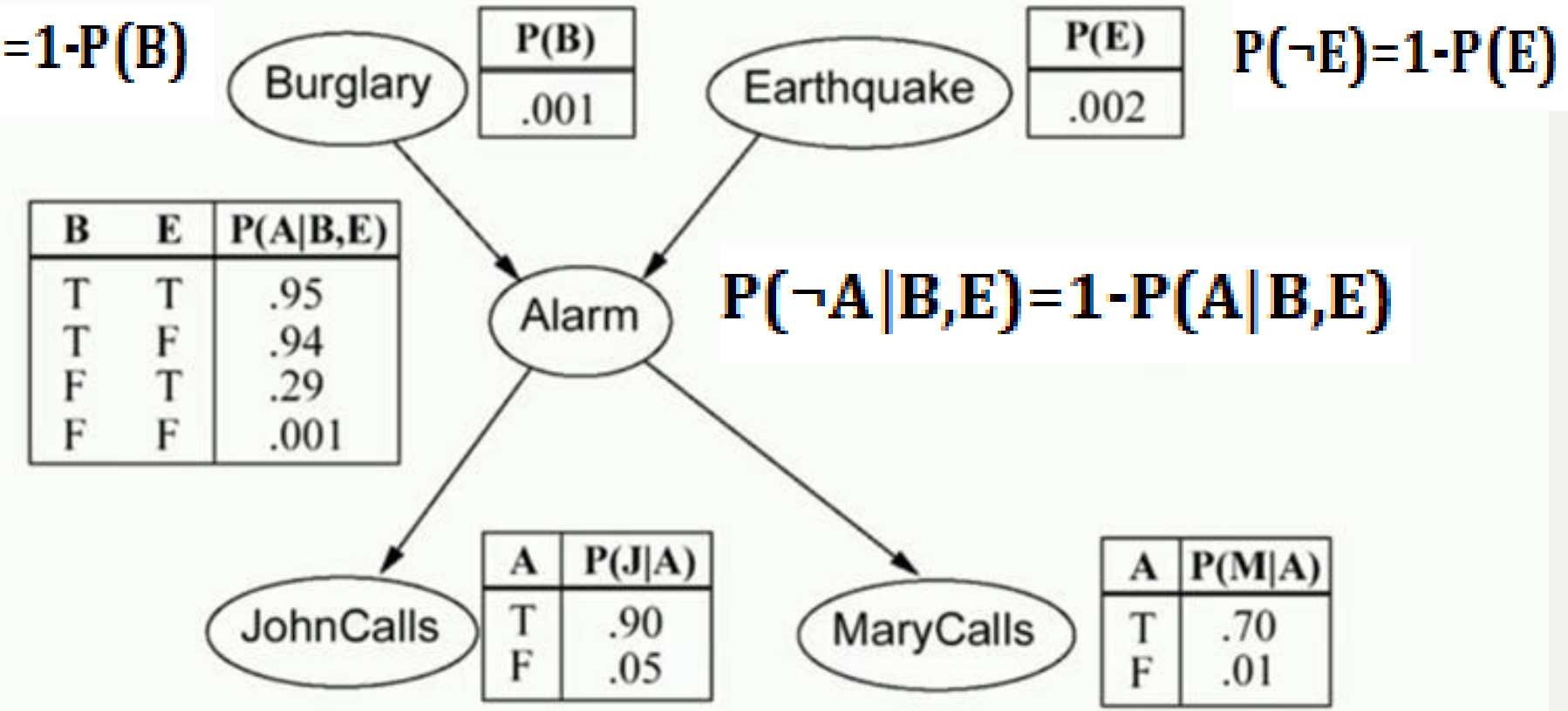
P(Wetgrass | Sprinkler, Rain) * P(Sprinkler | Cloudy) * P(Rain | Cloudy) * P (Cloudy)

BAYESIAN BELIEF NETWORKS – EXAMPLE – 1

- You have a new burglar alarm installed at home.
- It is fairly reliable at detecting burglary, but also sometimes responds to minor earthquakes.
- You have two neighbors, John and Merry , who promised to call you at work when they hear the alarm.
- John always calls when he hears the alarm, but sometimes confuses telephone ringing with the alarm and calls too.
- Merry likes loud music and sometimes misses the alarm.
- Given the evidence of who has or has not called, we would like to estimate the probability of a burglary.

BAYESIAN BELIEF NETWORKS – EXAMPLE – 1

$P(\neg B) = 1 - P(B)$

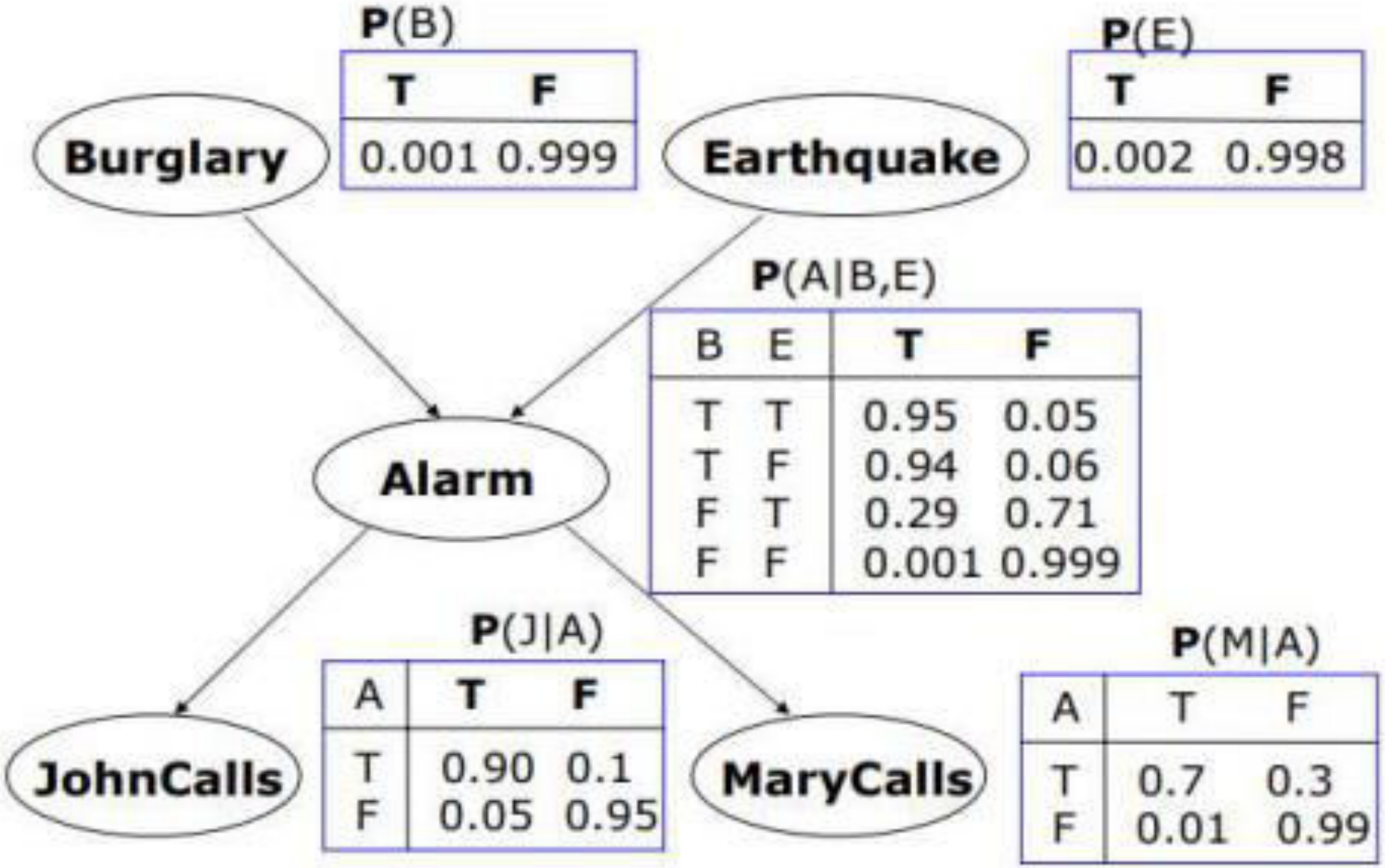


$P(\neg E) = 1 - P(E)$

$P(\neg A|B,E) = 1 - P(A|B,E)$

$P(\neg J|A) = 1 - P(J|A)$

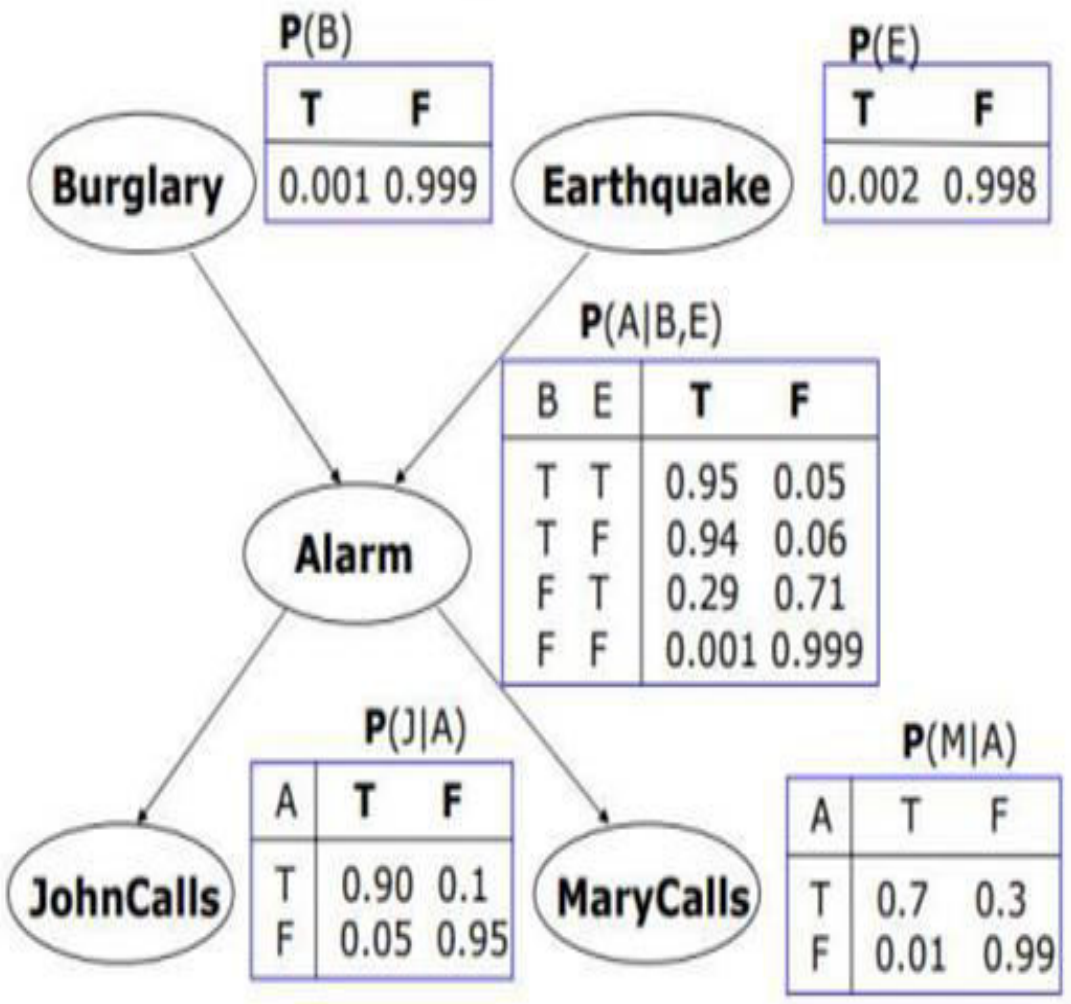
$P(\neg M|A) = 1 - P(M|A)$


P(B)
T F
0.001 0.999
Burglary
Earthquake
P(E)
T F
0.002 0.998
P(A|B,E)
B E T F
T T 0.95 0.05
T F 0.94 0.06
F T 0.29 0.71
F F 0.001 0.999
Alarm
P(J|A)
A T F
T 0.90 0.1
F 0.05 0.95
JohnCalls
MaryCalls
P(M|A)
A T F
T 0.7 0.3
F 0.01 0.99

BAYESIAN BELIEF NETWORKS – EXAMPLE – 1

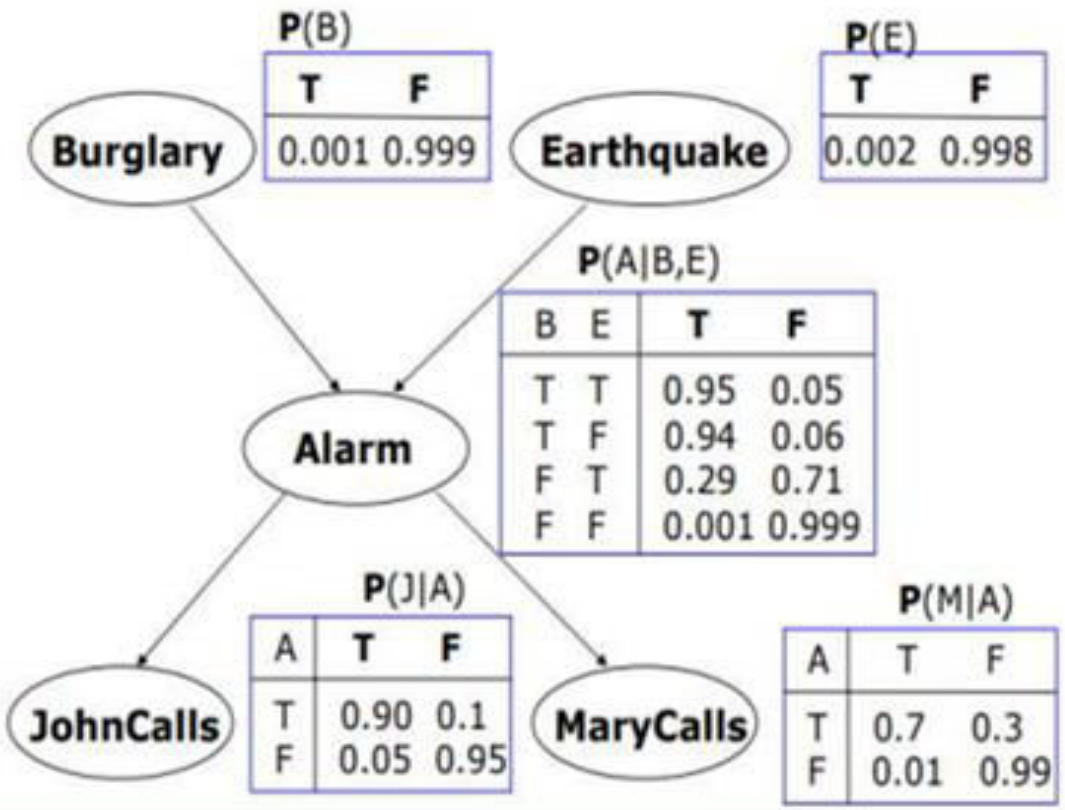
1. What is the probability that the alarm has sounded but neither a burglary nor an earthquake has occurred, and both John and Merry call?

Solution:



BAYESIAN BELIEF NETWORKS – EXAMPLE – 1

1. What is the probability that the alarm has sounded but neither a burglary nor an earthquake has occurred, and both John and Merry call?



Solution:

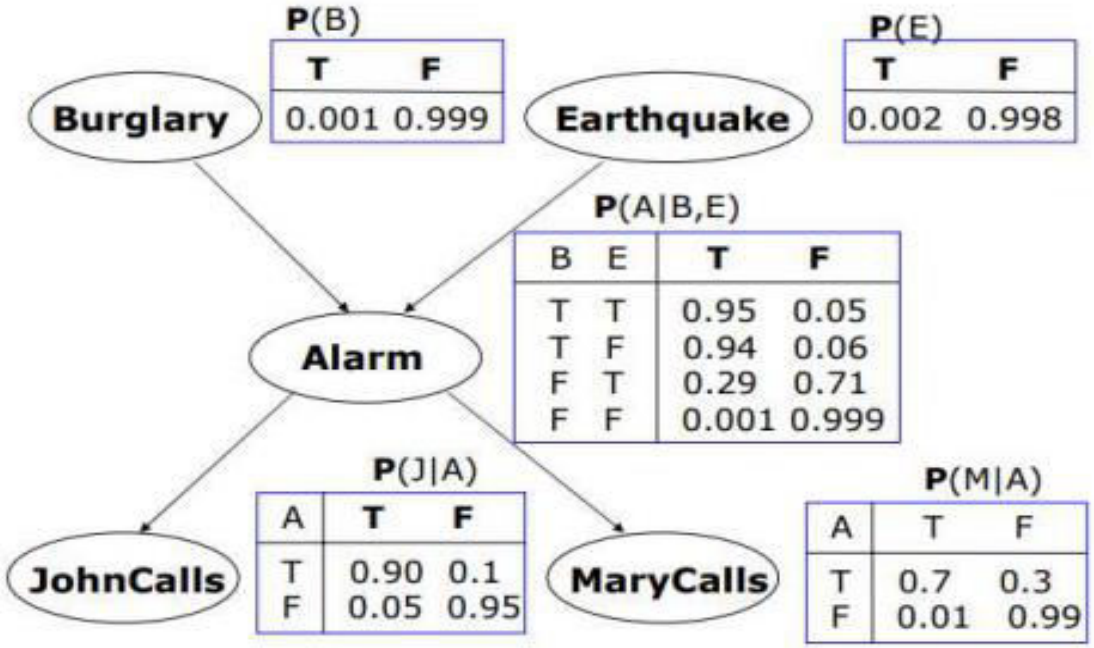
$$P(j \wedge m \wedge a \wedge \neg b \wedge \neg e) = P(j \mid a)\, P(m \mid a)\, P(a \mid \neg b, \neg e)\, P(\neg b)\, P(\neg e)$$

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

$$= 0.00062$$

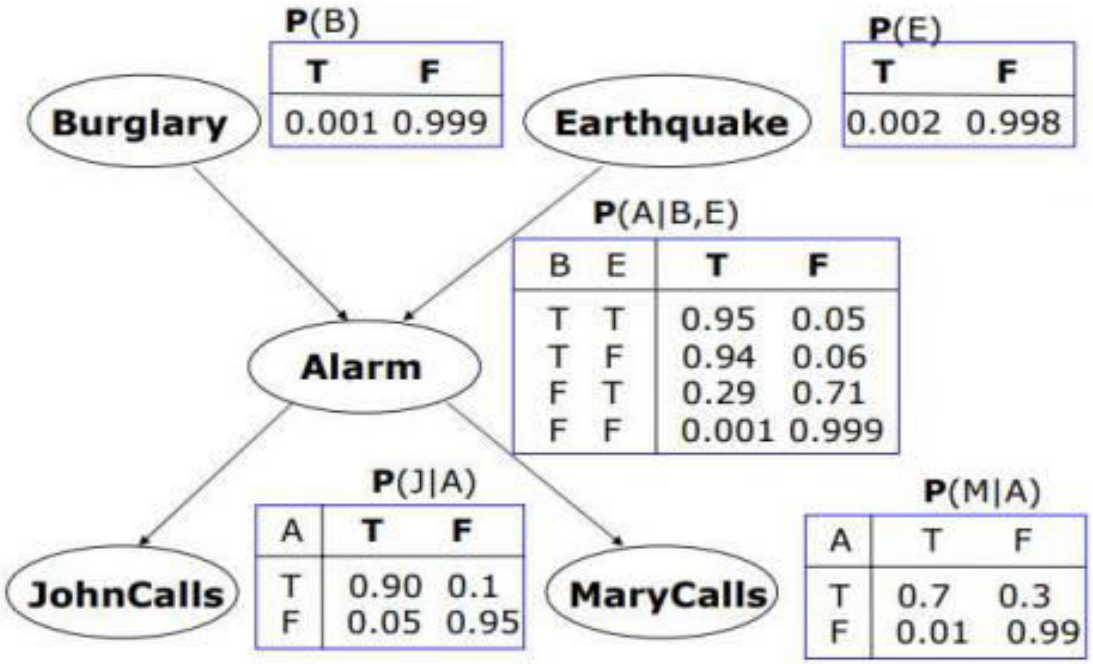
BAYESIAN BELIEF NETWORKS – EXAMPLE – 1

2. What is the probability that John call?



BAYESIAN BELIEF NETWORKS – EXAMPLE – 1

2. What is the probability that John call?



Solution:

$$P(j) = P(j \mid a)\,P(a) + P(j \mid \neg a)\,P(\neg a)$$

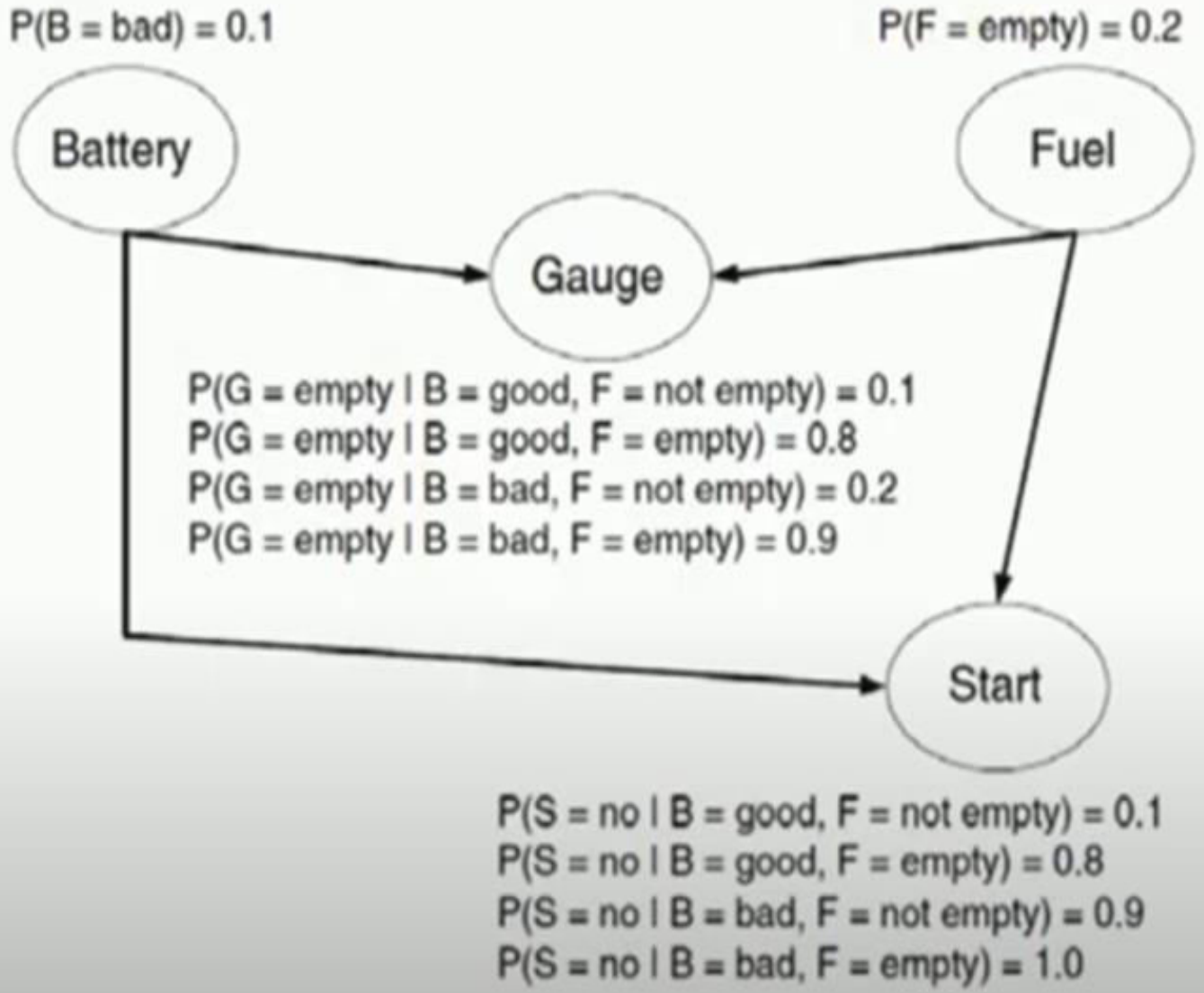
$$= P(j|a)\{P(a|b,e)*P(b,e)+P(a|\neg b,e)*P(\neg b,e)+P(a|b,\neg e)*P(b,\neg e)+P(a|\neg b,\neg e)*P(\neg b,\neg e)\}$$

$$+ P(j|\neg a)\{P(\neg a|b,e)*P(b,e)+P(\neg a|\neg b,e)*P(\neg b,e)+P(\neg a|b,\neg e)*P(b,\neg e)+P(\neg a|\neg b,\neg e)*$$

$$P(\neg b,\neg e)\}$$

$$= 0.90 * 0.00252 + 0.05 * 0.9974 = 0.0521$$

Bayesian Belief Network – Solved Example

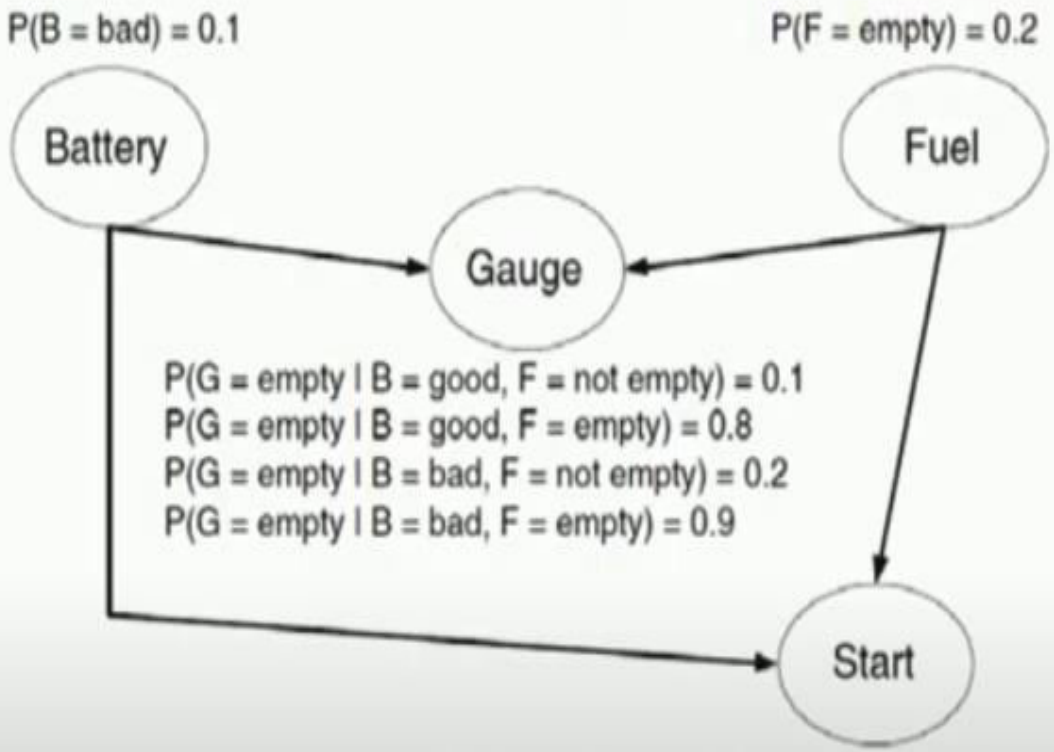


1. P(B=Good, F=Empty, G=Empty, S=Yes)
2. P(B=Bad, F=Empty, G=Not Empty, S=No)
3. Given the battery is bad, Compute the probability that the car will start

Bayesian Belief Network – Solved Example

1. **P(B=Good, F=Empty, G=Empty, S=Yes)**

$$
\begin{aligned}
& P(B = good, F = empty, G = empty, S = yes) \\
= \; & P(B = good) \times P(F = empty) \times P(G = empty | B = good, F = empty) \\
& \times P(S = yes | B = good, F = empty) \\
= \; & 0.9 \times 0.2 \times 0.8 \times 0.2 = 0.0288.
\end{aligned}
$$



2. **P(B=Bad, F=Empty, G=Not Empty, S=No)**

$$
\begin{aligned}
& P(B = bad, F = empty, G = not\ empty, S = no) \\
= \; & P(B = bad) \times P(F = empty) \times P(G = not\ empty | B = bad, F = empty) \\
& \times P(S = no | B = bad, F = empty) \\
= \; & 0.1 \times 0.2 \times 0.1 \times 1.0 = 0.002.
\end{aligned}
$$

3. Given the battery is bad, Compute the probability that the car will start.

$$p(S = yes \mid B = bad)$$

$$= p(S = yes, B = bad) \Big/ p(B = bad)$$

$$= \sum_{\alpha} p(S = yes, B = bad, F = \alpha) \Big/ p(B = bad)$$

$$= \sum_{\alpha} p(S = yes \mid B = bad, F = \alpha) \times p(B = bad, F = \alpha) \Big/ p(B = bad)$$

$$= \sum_{\alpha} p(S = yes \mid B = bad, F = \alpha) \times p(B = bad) \times p(F = \alpha) \Big/ p(B = bad)$$

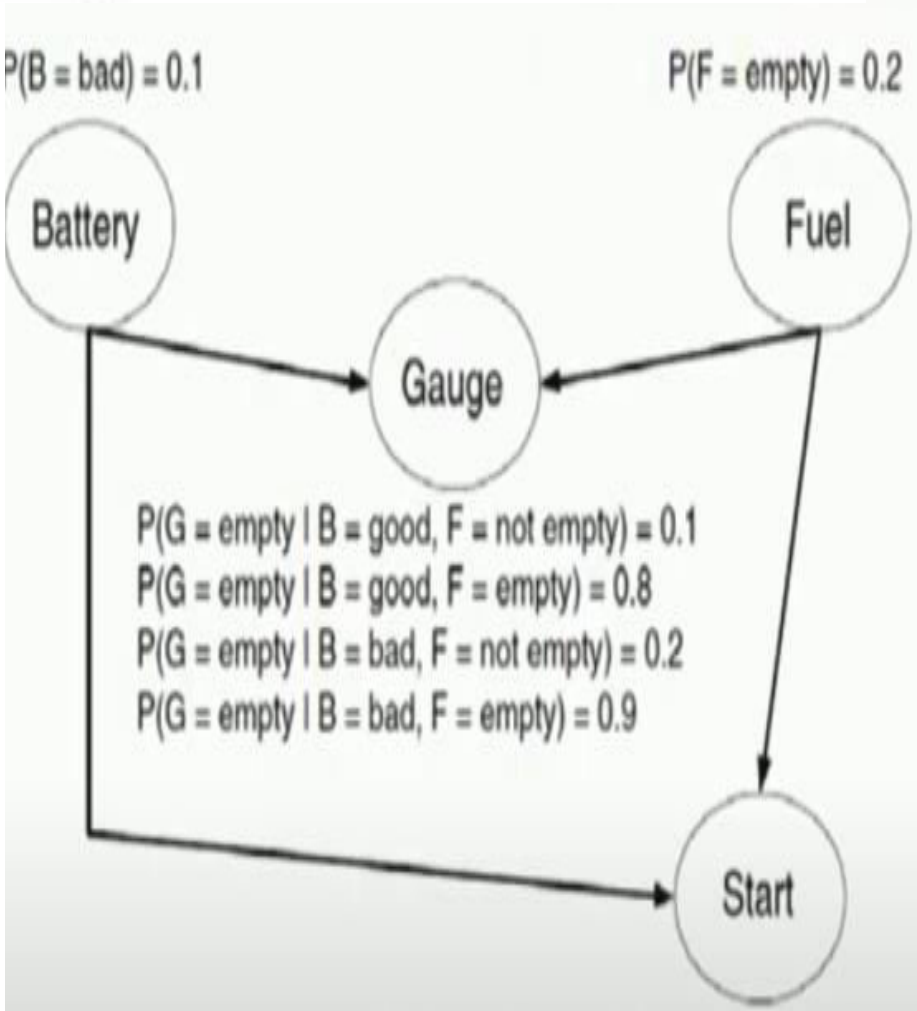
$$= \sum_{\alpha} p(S = yes \mid B = bad, F = \alpha) \times p(F = \alpha)$$

$$= P(S = Yes \mid B = bad, F = Empty) \cdot P(F = Empty) +$$

$$P(S = Yes \mid B = bad, F = Non - Empty) \cdot P(F = No - Empty)$$

$$= (0 \cdot 0.2) + (0.1 \cdot 0.8)$$

$$= 0.08$$



EM ALGORITHM

MOTIVATION

- In many practical learning settings, only a subset of the relevant instance features might be observable.
- For example, among many *Storm, Lightning, Thunder, ForestFire, Campfire,* and *BusTourGroup have been observed. (In BBN example)*
- If some variable is sometimes observed and sometimes not, then we can use the cases for which it has been observed to learn to predict its values when it is not.
- Many approaches have been proposed to handle the problem of learning in the presence of unobserved variables.
- The EM algorithm a widely used approach to learning in the presence of unobserved variables.
- The EM algorithm can be used
  - even for variables whose value is never directly observed,
  - provided the general form of the probability distribution governing these variables is known.

Estimating Means of k Gaussians

- Consider a problem in which the data D is a set of instances are - a mixture of k distinct Normal distributions.
- This problem setting is illustrated in Figure for the case where k = 2 and where the instances are the points shown along the x axis.
- Each instance is generated using a two-step process.
  - First, one of the k Normal distributions is selected at random.
  - Second, a single random instance x_i is generated according to this selected distribution.
- This process is repeated to generate a set of data points as shown in the figure.

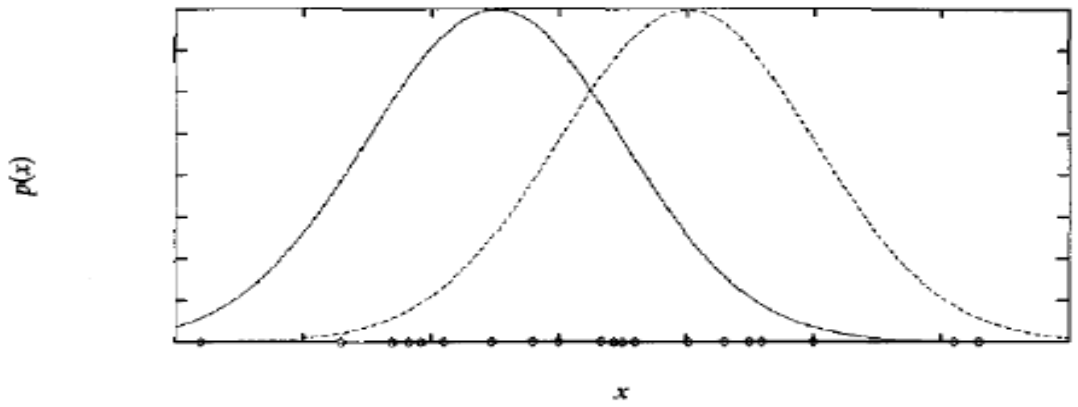
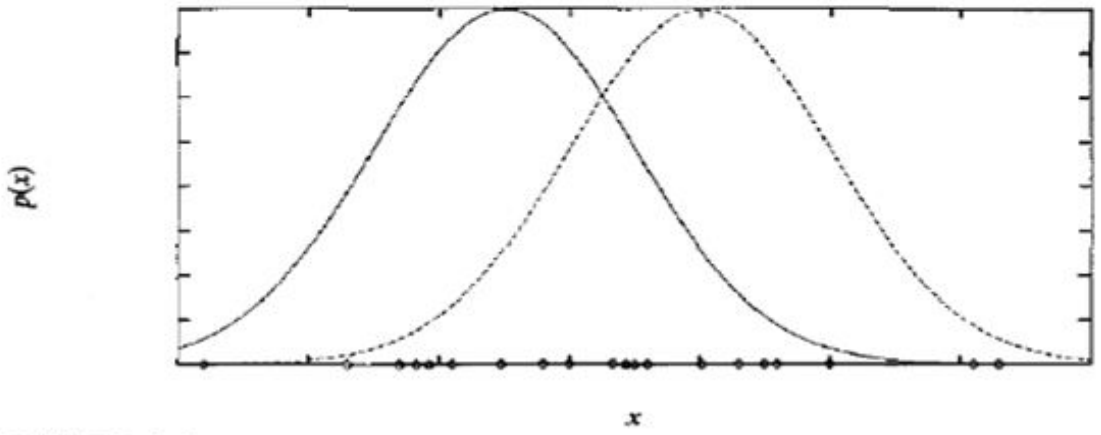


FIGURE 6.4
Instances generated by a mixture of two Normal distributions with identical variance σ. The instances are shown by the points along the x axis. If the means of the Normal distributions are unknown, the EM algorithm can be used to search for their maximum likelihood estimates.

Estimating Means of k Gaussians

- To simplify our discussion, we 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, known value.
- The learning task is to output a hypothesis $h = (\mu_1, \ldots, \mu_k)$ 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).



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

In this case, the sum of squared errors is minimized by the sample mean

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

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.

Estimating Means of k Gaussians

- Our problem here, however, involves a mixture of k different Normal distributions, and we cannot observe which instances were generated by which distribution.
- we can think full description of each instance as the triple **(x_i, z_{i1}, z_{i2})**,
  - where x_i is the observed value of the i^{th} instance and
  - where z_{i1} and z_{i2} indicate which of the two Normal distributions was used to generate the value x_i.
- In particular, z_{ij} has the value 1 if x_i was created by the j^{th} 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, we could use following Equation to solve for the means μ_1 and μ_2

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

  - Because they are not, we will instead use the EM algorithm.

Estimating Means of k Gaussians

•Applied to the problem of estimating the two means for Figure, the EM algorithm first initializes the hypothesis to h = **(μ1, μ2),** where **μ1** and **μ2** 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**.

Estimating Means of k Gaussians

Step 1: 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.

Step 2: 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 in Step 1. Then replace the hypothesis $h = \langle \mu_1, \mu_2 \rangle$ by the new hypothesis $h' = \langle \mu'_1, \mu'_2 \rangle$ and iterate.

Let us examine how both of these steps can be implemented in practice.

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

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

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

Thus the first step is implemented by substituting the current values $\langle \mu_1, \mu_2 \rangle$ and the observed x_i into the above expression.

Estimating Means of k Gaussians

— In the second step we use the $E[z_{ij}]$ calculated during Step 1 to derive a new maximum likelihood hypothesis $h' = \langle \mu'_1, \mu'_2 \rangle$.

— The maximum likelihood hypothesis in this case is given by

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

- This expression is similar to the sample mean from Equation that is used to estimate **μ** for a single Normal distribution.

- Our new expression is just the weighted sample mean for **μj,** *with each instance weighted by the expectation* ***E[z,j]*** *that it was generated by the* ***jth*** *Normal distribution*

Estimating Means of k Gaussians

- The above algorithm for estimating the means of a mixture of k Normal distributions illustrates the essence of the EM approach:
  - The current hypothesis is used to estimate the unobserved variables, and the expected values of these variables are then used to calculate an improved hypothesis.
  - It can be proved that on each iteration through this loop, the EM algorithm increases the likelihood **P(D|h)** unless it is at a local maximum.
  - The algorithm thus converges to a local maximum likelihood hypothesis for **(μl, *μ2).***

Estimating Means of k Gaussians

- The EM algorithm can be applied in many settings where we wish to estimate some set of parameters θ that describe an underlying probability distribution, given only the observed portion of the full data produced by this distribution.
- For example the parameters of interest were θ = ***(μ1, μ2),*** and the full data were the triples ***(xi, zi1, zi2)*** of which only the ***xi*** were observed.
- In general let **X = *{x1,*** . . . , ***xm}*** denote the observed data in a set of m independently drawn instances, let **Z = *{z1,*** . . . , ***zm}*** denote the unobserved data in these same instances, and let ***Y*** **= X U Z** denote the full data.
- Note the unobserved **Z** can be treated as a random variable whose probability distribution depends on the unknown parameters θ and on the observed data X.
- Similarly, ***Y*** is a random variable because it is defined in terms of the random variable Z.
- We use ***h*** to denote the current hypothesized values of the parameters θ, and ***h'*** to denote the revised hypothesis that is estimated on each iteration of the EM algorithm.

Estimating Means of k Gaussians

- The EM algorithm searches for the maximum likelihood hypothesis **h'** by seeking the **h'** that maximizes **E[ln P(Y |h')]**.
- This expected value is taken over the probability distribution governing **Y** , which is determined by the unknown parameters θ.
- Let us consider exactly what this expression signifies.
- First, **P(Y|h')** is the likelihood of the full data **Y** given hypothesis **h'.** It is reasonable that we wish to find a **h'** that maximizes some function of this quantity.
- Second, maximizing the logarithm of this quantity **ln P(Y|h')** also maximizes **P(Yh'),** as we have discussed on several occasions already.
- Third, we introduce the expected value **E[ln P(Yh')]** because the full data **Y** is itself a random variable.

Estimating Means of k Gaussians

- Given that the full data **Y** is a combination of the observed data X and unobserved data Z, we must average over the possible values of the unobserved Z, weighting each according to its probability.
- In other words we take the expected value **E[ln P(Y|h')]** over the probability distribution governing the random variable **Y**
- The distribution governing **Y** is determined by the completely known values for X, plus the distribution governing Z.
- What is the probability distribution governing **Y?**
- In general we will not know this distribution because it is determined by the parameters θ that we are trying to estimate.
- Therefore, the EM algorithm uses its current hypothesis **h** in place of the actual parameters θ to estimate the distribution governing **Y**.
- Let us define a function **Q(h' | h)** that gives **E[ln P(Y |h')]** as a function of **h'**, under the assumption that θ = **h** and given the observed portion X of the full data **Y**.

$$Q(h'|h) = E[\ln p(Y|h')|h, X]$$

Estimating Means of k Gaussians

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

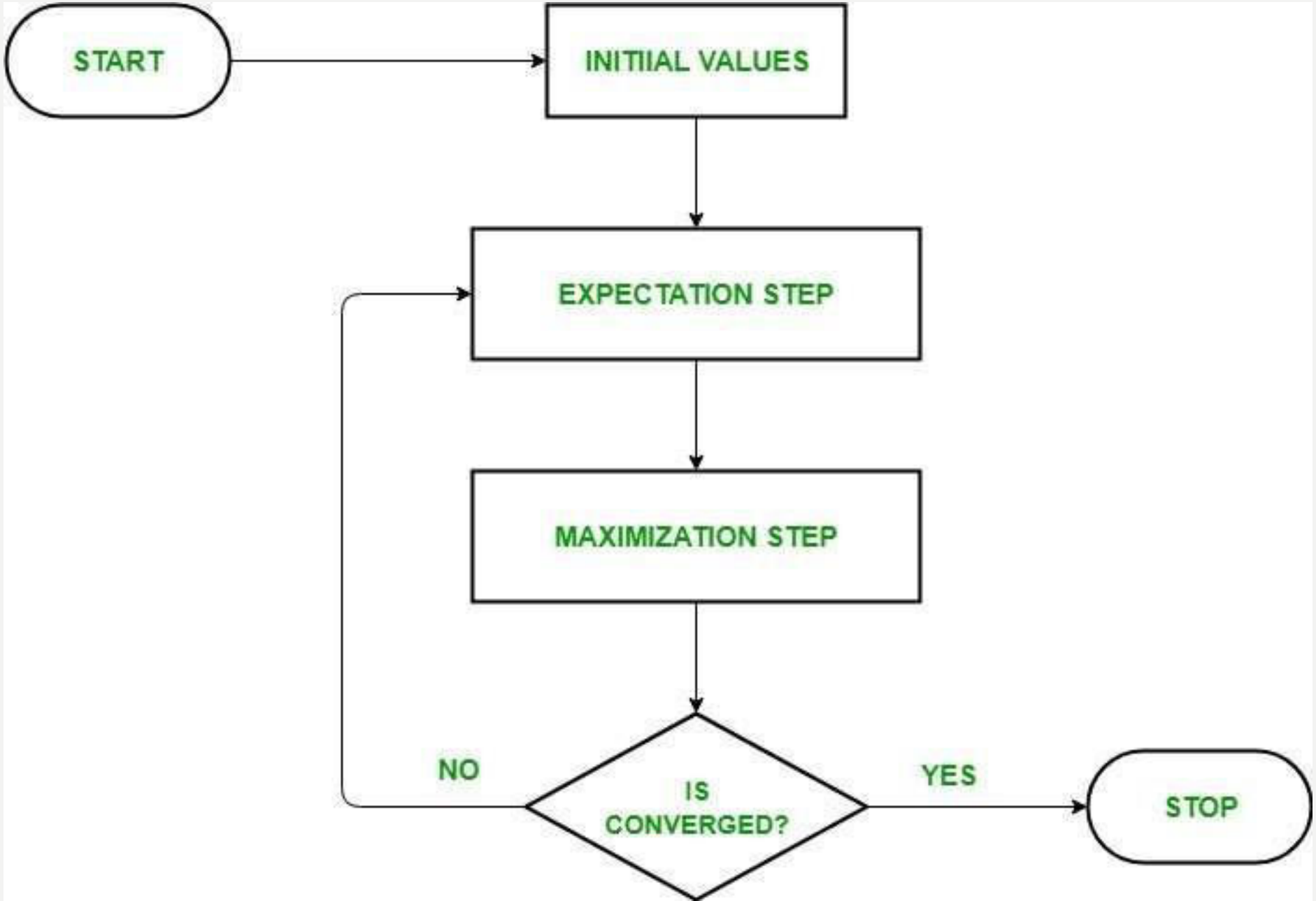
- **Step 1: *Estimation* (E) *step:*** Calculate ***Q(h' | h)*** using the current hypothesis ***h*** and the observed data X to estimate the probability distribution over ***Y.***

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

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

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

EM ALGORITHM



EM ALGORITHM

Advantages of EM algorithm –

- It is always guaranteed that likelihood will increase with each iteration.
- The E-step and M-step are often pretty easy for many problems in terms of implementation.
- Solutions to the M-steps often exist in the closed form.

Disadvantages of EM algorithm –

- It has slow convergence.
- It makes convergence to the local optima only.
- It requires both the probabilities, forward and backward (numerical optimization requires only forward probability).

① **E-step**
↳ the missing data are estimated given the observed data & current estimate of the model parameter

② **M-step**
↳ the likelihood function is maximized under the assumption that the missing data are known. The estimate of the missing data from the E-step are used in lieu of the actual missing data

E-step

$x = \{1, 8, 12, 7, -, -\}$

$n = 6 \quad \bar{x}(\mu_0) = \frac{38}{6} = 4.67 \approx ⑤$

$x = \{1, 8, 12, 7, 5, 5\}$

$\bar{x}_1(\mu_1) = 4.67 + \frac{5+5}{6} = 4.67 + 1.67 = 6.34 \quad D_1 = (6.78 - 6.39) = 0.44$

$\bar{x}_2(\mu_2) = 4.67 + \frac{6.34+6.34}{6} = 4.67 + 2.11 = 6.78 \quad D_2 = (6.93 - 6.78) = 0.15$

$\bar{x}_3(\mu_3) = 4.67 + \frac{6.78+6.78}{6} = 4.67 + 2.26 = 6.93 \quad D_3 = (6.98 - 6.93) = 0.05$

$\bar{x}_4(\mu_4) = 4.67 + \frac{6.93+6.93}{6} = 4.67 + 2.31 = 6.98$

$x = \{1, 8, 12, 7, 6.98, 6.98\}$ from E-step

M-step:-

Likelihood $(x|\theta)$

$= L(x|\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$

$\log L(x|\theta) = -\sum_{i=1}^{n} \frac{(x_i-\mu)^2}{2\sigma^2} - 0.5n\log 2\pi - n\log\sigma$

objective is to identify :- μ & σ given X (data) that maximize eqn.

$x_i \in X$, find $(\mu_1, \sigma_1), (\mu_2, \sigma_2), \ldots (\mu_K, \sigma_K) \ldots (\mu_n, \sigma_n)$ → $\log L(x|\theta)$ is max

eg:- $x = 1, 2, 3, 4, 5$

σ	μ	$\log L(x\|\theta)$
1	1	-16.995
2	2	-5.376
2	3	-4.751 } max
3	2	-5.214
3	3	-4.937

Hence take $\mu = 2, 3$

K-MEANS ALGORITHM FOR CLUSTERING

- kMeans algorithm is an unsupervised learning algorithm
- Given a data set of items, with certain features, and values for these features, the algorithm will categorize the items into k groups or clusters of similarity.
- To calculate the similarity, we use the Euclidean distance, Manhattan distance, Hamming distance, Cosine distance as measurement.

K-MEANS ALGORITHM FOR CLUSTERING

Here is the pseudocode for implementing a K-means algorithm.

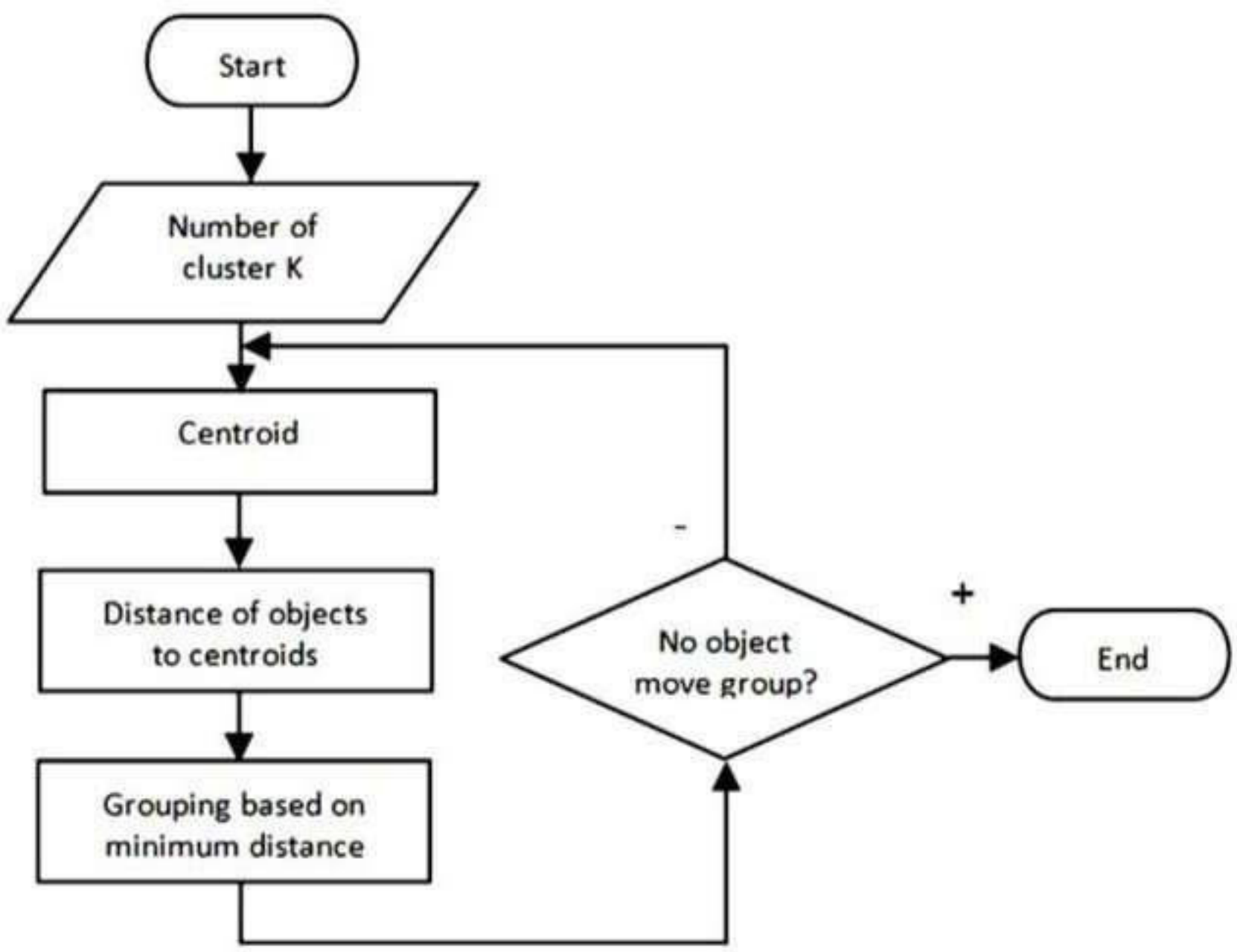
Input: Algorithm K-Means (K number of clusters, D list of data points)

1. Choose K number of random data points as initial centroids (cluster centers).

2. Repeat till cluster centers stabilize:
    a. Allocate each point in D to the nearest of Kth centroids.

    b. Compute centroid for the cluster using all points in the cluster.

K-MEANS ALGORITHM FOR CLUSTERING



ADVANTAGES AND DISADVANTAGES OF K-MEANS ALGORITHM

Advantages of K-Means Algorithm

1. K-means algorithm is simple, easy to understand, and easy to implement.
2. It is also efficient, in which the time taken to cluster K-means rises linearly with the number of data points.
3. No other clustering algorithm performs better than K-means.

3.

Disadvantages of K-Means Algorithm

1. The user needs to specify an initial value of K.
2. The process of finding the clusters may not converge.
3. It is not suitable for discovering clusters that are not hyper ellipsoids or hyper spheres).

Derivation of the k Means Algorithm

- The k-means problem is to estimate the parameters **θ** = ***(μ1, μ2)*** that define the means of the k Normal distributions. We are given the observed data X = ***{ < xi > }*** .
- The hidden variables **Z** = ***{ < zi1 ,. . . , zik>}*** in this case indicate which of the k Normal distributions was used to generate ***xi.***
- To apply EM we must derive an expression for ***Q(h|h')*** that applies to our k-means problem.
- First, let us derive an expression for ***ln p(Y|h'),***
- Note the probability ***p(yi |h')*** of a single instance ***yi*** = ***(xi,zi1 , . . .zik)*** **o**f the full data can be written,

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

Derivation of the k Means Algorithm

- Given this probability for a single instance ***p(yi|h'),*** the logarithm of the probability ln ***P(Y|h')*** for all ***m*** instances in the data is,

$$\ln P(Y|h') = \ln \prod_{i=1}^{m} p(y_i|h')$$

$$= \sum_{i=1}^{m} \ln p(y_i|h')$$

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

Derivation of the k Means Algorithm

- Finally we must take the expected value of this ln P(Y|h') over the probability distribution governing Y

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

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

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

Derivation of the k Means Algorithm

- Thus, the first (estimation) step of the EM algorithm defines the Q function based on the estimated E[zij] terms.
- The second (maximization) step then finds the values $\mu_1', \ldots. \mu_n'$ that maximize this Q function.
- In the current case

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

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

- Thus, the maximum likelihood hypothesis here minimizes a weighted sum of squared errors, where the contribution of each instance xi to the error that defines μ_j' is weighted by E[zij].

K-means

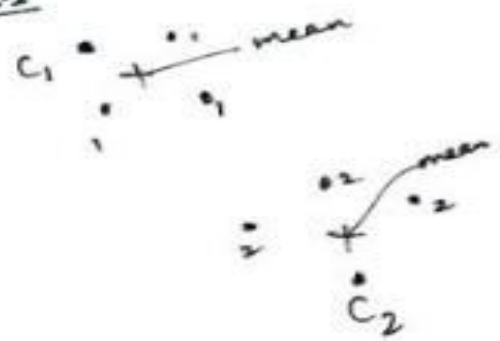
This is a prototype based, partitional clustering technique that attempt to find a user specified no. of clusters K, w/c are represented by their centroids

Algm

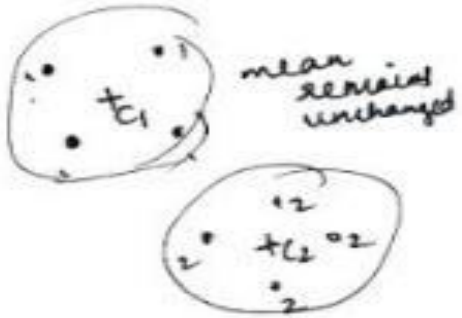
i/p: Data, K
o/p: cluster(Cs);

1. Randomly select k point as initial cluster center
2. (re) assign each data to its nearest cluster center
3. (re) calculate means;
4. repeat step 2 & 3 until the convergence criteria is satisfied

Iter-1



Iter-2



$$MSSE = \frac{1}{n}\sum_{j=1}^{K}\sum_{i=1}^{|C_j|} dist(x_i, c_j)$$

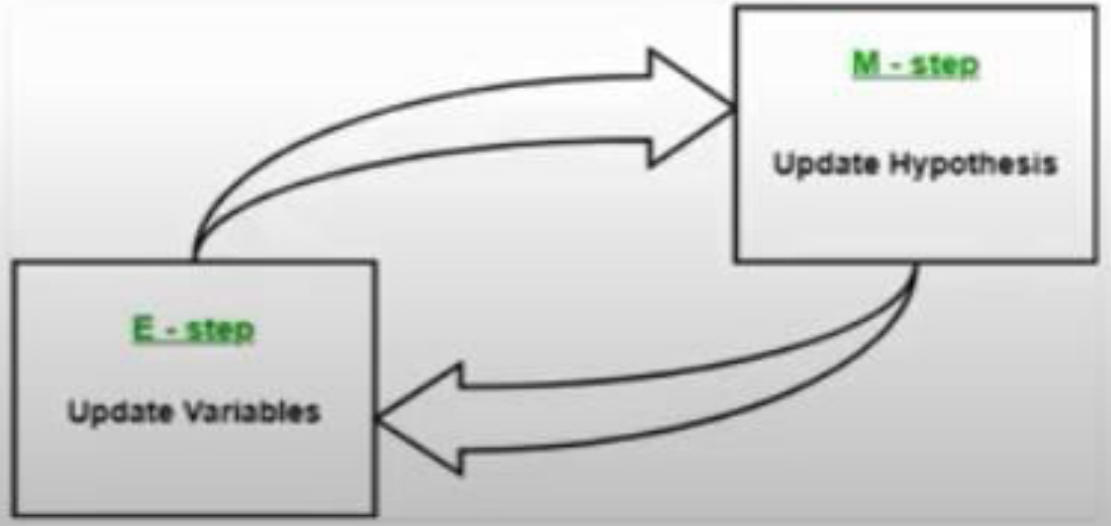
ML Lab Program: 8

- TITLE: **CLUSTERING BASED ON EM ALGORITHM AND K-MEANS**

- Apply **EM algorithm** to cluster a set of data stored in a .CSV file. Use the same data set for **clustering using k-Means algorithm.** Compare the results of these two algorithms and comment on the **quality of clustering.** You can add Java/Python ML library classes/API in the program.

What is EM algorithm in machine learning?

- The **Expectation-Maximization algorithm** is an approach for performing maximum likelihood estimation in the presence of latent variables. It does this by first estimating the values for the latent variables, then optimizing the model, then repeating these two steps until convergence.



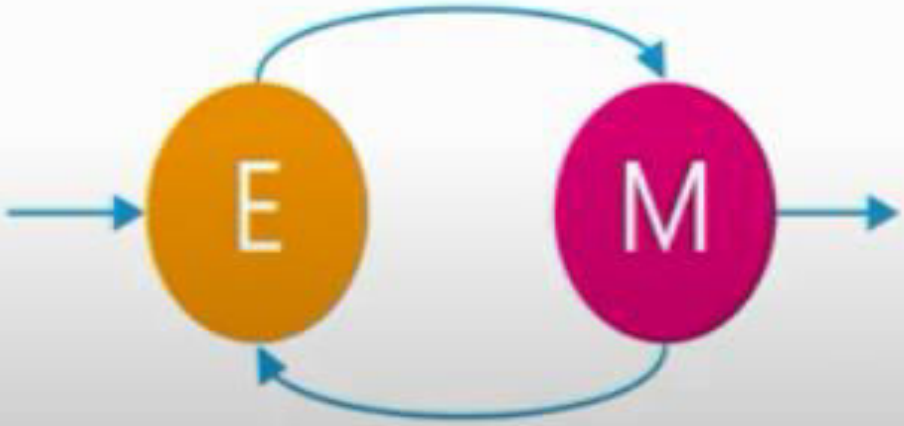
Gaussian Mixture Models(GMM)

- Gaussian Mixture Models are probabilistic models and use the soft clustering approach for distributing the points in different clusters.
- *Gaussian Mixture Models (GMMs) assume that there are a certain number of Gaussian distributions, and each of these distributions represent a cluster. Hence, a Gaussian Mixture Model tends to group the data points belonging to a single distribution together.*

Expectation-Maximization Algorithm

Algorithm:

1. Given a set of incomplete data, consider a set of starting parameters.
2. **Expectation step (E – step):** Using the observed available data of the dataset, estimate (guess) the values of the missing data.
3. **Maximization step (M – step):** Complete data generated after the expectation (E) step is used in order to update the parameters.
4. Repeat step 2 and step 3 until convergence.



ALGORITHM

K-Means Algorithm

- **K-Means** algorithm is an iterative algorithm that tries to partition the dataset into *K* pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to **only one group**.

- It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible.

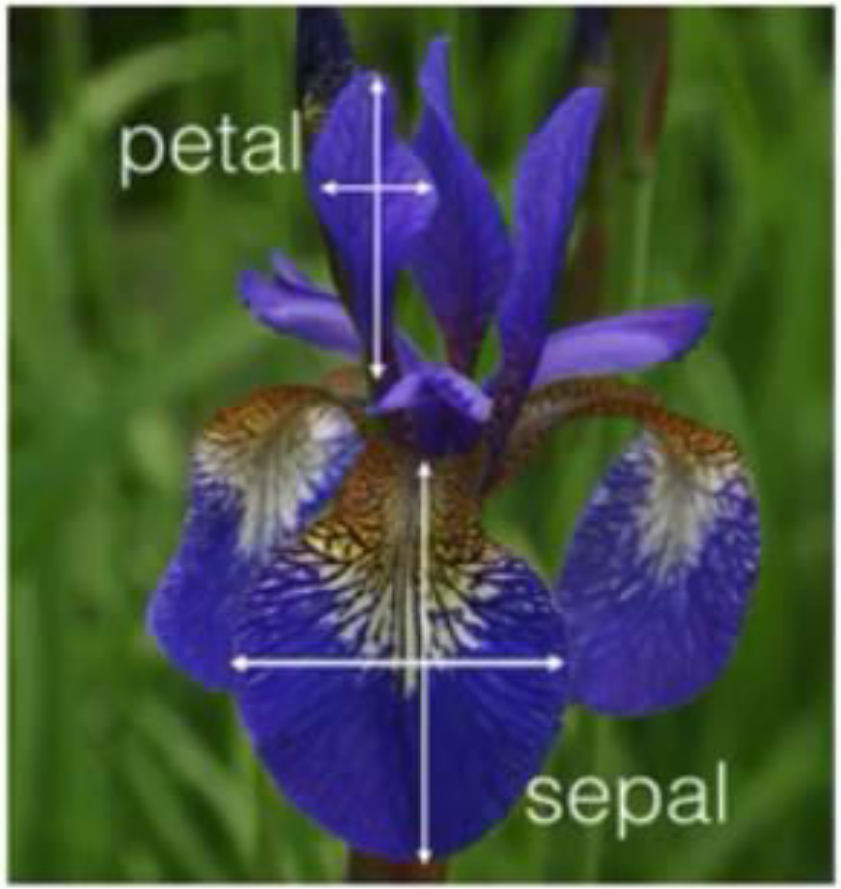
- It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum.

Algorithm 1 *k*-means algorithm

1: Specify the number k of clusters to assign.
2: Randomly initialize k centroids.
3: **repeat**
4: **expectation:** Assign each point to its closest centroid.
5: **maximization:** Compute the new centroid (mean) of each cluster.
6: **until** The centroid positions do not change.

IRIS data set

- This data sets consists of 3 different types of irises' (Setosa, Versicolour, and Virginica) petal and sepal length, stored in a 150x4 numpy.ndarray
- The rows being the samples and the columns being: Sepal Length, Sepal Width, Petal Length and Petal Width.

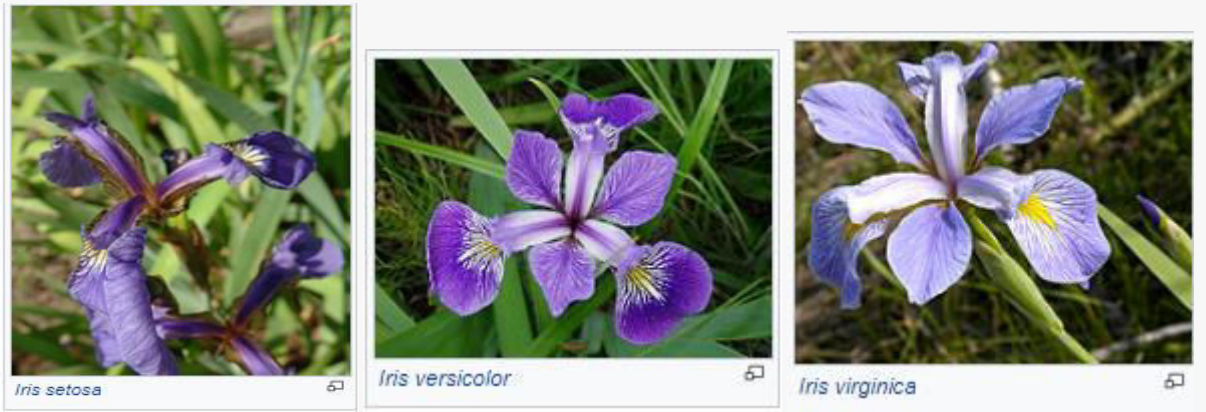


The iris dataset is a classic and very easy multi-class classification dataset.

Classes	3
Samples per class	50
Samples total	150
Dimensionality	4
Features	real, positive

PROGRAM

In [11]:
```
import matplotlib.pyplot as plt
from sklearn import datasets

import sklearn.metrics as sm
import pandas as pd
import numpy as np

iris = datasets.load_iris()
X = pd.DataFrame(iris.data)
X.columns = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']

y = pd.DataFrame(iris.target)
y.columns = ['Targets']



plt.figure(figsize=(14,7))
colormap = np.array(['red', 'lime', 'black'])
```



PROGRAM

```
# Plot the Models Classifications
#K-Means
from sklearn.cluster import KMeans
model = KMeans(n_clusters=3)
model.fit(X)
y_km=model.predict(X)
plt.subplot(1, 2, 2)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y_km], s=40)
plt.title('K Mean Classification')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
print('The accuracy score of K-Mean: ',sm.accuracy_score(y, y_km))
print('The Confusion matrixof K-Mean: \n',sm.confusion_matrix(y, y_km))
```

PROGRAM

```
#EM
from sklearn.mixture import GaussianMixture
gmm = GaussianMixture(n_components=3)
gmm.fit(X)
y_gmm = gmm.predict(X)
plt.subplot(1, 2, 1)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y_gmm], s=40)
plt.title('GMM Classification')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
print('The accuracy score of EM: ',sm.accuracy_score(y, y_gmm))
print('The Confusion matrix of EM: \n',sm.confusion_matrix(y, y_gmm))
```

```
The accuracy score of EM:  0.9666666666666667
The Confusion matrix of EM:
 [[50  0  0]
 [ 0 45  5]
 [ 0  0 50]]
```

```
The accuracy score of K-Mean:  0.09333333333333334
The Confusion matrixof K-Mean:
 [[ 0 50  0]
 [ 2  0 48]
 [36  0 14]]
```