S. J. P. N. TRUST'S

# HIRASUGAR INSTITUTE OF TECHNOLOGY, NIDASOSHI

Accredited at 'A' Grade by NAAC

Programmes Accredited by NBA: CSE, ECE, EEE & ME.

## Department of Computer Science & Engineering

## Course: Artificial Intelligence Machine Learning (18CS71)



# Module 4: Bayesian Learning

Dr. Mahesh G. Huddar

Asst. Prof. , Dept. of Computer Science & Engg.,

Hirasugar Institute of Technology, Nidasoshi

# Bayesian Classification: Why? - Features

- Each observed training example can incrementally decrease or increase the estimated probability that a hypothesis is correct. This provides a more flexible approach to learning than algorithms that completely eliminate a hypothesis if it is found to be inconsistent with any single example.



- Prior knowledge can be combined with observed data to determine the final probability of a hypothesis. In Bayesian learning, prior knowledge is provided by asserting

1. a prior probability for each candidate hypothesis, and
2. a probability distribution over observed data for each possible hypothesis.

# Bayesian Classification: Why? - Features

- Bayesian methods can accommodate hypotheses that make probabilistic predictions (e.g., hypotheses such as "this pneumonia patient has a 93% chance of complete recovery").
- New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities.
- Even in cases where Bayesian methods prove computationally intractable, they can provide a standard of optimal decision making against which other practical methods can be measured.

# Practical Difficulty

- Bayesian methods 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.
- Computational cost required to determine the Bayes optimal hypothesis in the general case.

# Naïve Bayes Model – Simple Classification Example

- Suppose a **XYZ** company needs to predict the service required by the incoming customer.
- If there are only two services offered — R and M.
- Then the value to be predicted is whether the next customer will be for R or M.
- The number of classes k are 2 that is, k=2.

# Naïve Bayes Model – Simple Classification Example

- The first step is to compute prior probability.
- Suppose the data gathered for the last one year showed that during that period there were 2500 customers for R and 1500 customers for M.



- Thus, prior probability for the next customer to be for R is 2500/4000 or 5/8.
- Prior probability for the next customer to be for M is 1500/4000 or 3/8.
- Based on this information alone, the next customer would likely be for R.

# Naïve Bayes Model – Simple Classification Example

- Another way to predict the service requirement by the next customer is to look at the most recent data.
- One can look at the last few (choose a number) customers, to predict the next customer.



- Suppose the last five customers were for the services ... R, M, R, M, M order.
- Thus, the data shows the recent probability of R is 2/5 and that of M is 3/5.
- Based on just this information, the next customer will likely to be for M.

# Naïve Bayes Model – Simple Classification Example

- Thomas Bayes suggested that the prior probability should be informed by the more recent data.
- Naive-Bayes posterior probability for a class is computed by multiplying the prior probability and the recent probability.
- NB posterior probability P(R) is (5/8 x 2/5) = 10/40.
- Similarly, the NB probability P(M) is (3/8 x 3/5) = 9/40.
- Since P(R) is greater than P(M), it follows that there is a greater probability of the next customer to be for R.
- Thus the expected class label assigned to the next customer would be R.

# Naïve Bayes Model – Simple Classification Example

- Suppose, however the next customer coming in was for M service.
- The last five customer sequence now becomes M, R, M, M, M.
- Thus, the recent data shows the probability for R to be 1/5 and that of M to be 4/5.
- Now the NB probability for R is (5/8 x 1/5) = 5/40.
- Similarly, the NB probability for M is (3/8 x 4/5) = 12/40.
- Since P(M) is greater than P(R), it follows that there is a greater probability of the next customer to be for M.
- Thus the expected class label assigned to the next customer is M.

# BAYES THEOREM

- In machine learning we are often interested in determining the best hypothesis from some space H, given the observed training data D.
- One way to specify what we mean by the **best hypothesis is to say that we demand the most probable** hypothesis, given the data **D plus any initial knowledge about the prior probabilities** of the various hypotheses in H.
- Bayes theorem provides a direct method for calculating such probabilities.
- More precisely, 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

- To define Bayes theorem precisely, let us first introduce a little notation.
- We shall write **P(h) to denote the initial probability that hypothesis h holds, before we** have observed the training data. **P(h) is often called the priorprobability of h and** may reflect any background knowledge we have about the chance that **h is a correct** hypothesis.
- If we have no such prior knowledge, then we might simply assign the same prior probability to each candidate hypothesis.

# BAYES THEOREM

- **P(D) to denote the prior probability that training data D will be observed (i.e.,** the probability of D given no knowledge about which hypothesis holds).
- Next, we will write **P(D|h) to denote the probability of observing data D given some** world in which hypothesis **h holds. More generally, we write P(x|y) to denote** the probability of x given **y.**
- **In machine learning problems we are interested in** the probability **P (h|D) that h holds given the observed training data D.**
- **P (h|D) is** called the **posteriorprobability of h, because it reflects our confidence that h holds** after we have seen the training data **D.**
- **Notice the posterior probability P(h|d)** reflects the influence of the training data **D, in contrast to the prior probability P(h) , which is independent of D.**

# BAYES THEOREM

- 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).**



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

# BAYES THEOREM

- The learner considers some set of candidate hypotheses H and is interested in finding the most probable hypothesis **h ϵ H** given the observed data **D (or at least one of the maximally probable if there** are several).
- Any such maximally probable hypothesis is called a **maximum a posteriori (MAP) hypothesis.**
- **We can determine the MAP hypotheses by using** Bayes theorem to calculate the posterior probability of each candidate hypothesis.

# BAYES THEOREM

- More precisely, we will say that $h_{MAP}$ **is a MAP hypothesis provided**

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

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

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



- Notice in the final step above we dropped the term **P(D) because it is a constant** independent of **h.**

# BAYES THEOREM

- In some cases, we will assume that every hypothesis in H is equally probable a priori **(P(hi) = P(hj) for all hi and hj in H).**

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

# Example - Does patient have cancer or not?

- Consider a medical diagnosis problem in which there are two alternative hypotheses:

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

  (2) that the patient does not.

- The available data is from a particular laboratory test with two possible outcomes: + (positive) and − *(negative).*
- 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 we know that, 0.008 of the entire population have cancer.
- $P(cancer) = 0.008$ $P(\neg cancer) = 0.992$
- $P(+|cancer) = 0.98$ $P(-|cancer) = 0.02$
- $P(+|\neg cancer) = 0.03$ $P(-|\neg cancer) = 0.97$

# Example - Does patient have cancer or not?

- Suppose we now observe a new patient for whom the lab test returns a **positive** result.
- Should we diagnose the patient as having cancer or not?



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

$$P(cancer|+) = P(+|cancer) * P(cancer) = 0.98 \ * 0.008 = 0.0078$$

$$P(\neg cancer|+) = P(+|\neg cancer) * P(\neg cancer) = 0.03 \ * 0.992 = 0.0298$$

$$h_{MAP} = \neg cancer$$

# Example - Does patient have cancer or not?

- Suppose we now observe a new patient for whom the lab test returns a **negative** result.
- Should we diagnose the patient as having cancer or not?



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

$$P(cancer|-) = P(-|cancer) * P(cancer) = 0.02 \; * 0.008 = 0.00016$$

$$P(\urcorner cancer|-) = P(-|\urcorner cancer) * P(\urcorner cancer) = 0.97 \; * 0.992 = 0.96224$$

$$h_{MAP} = \urcorner cancer$$

# Brute-Force Bayes Concept Learning

- Assume the learner considers some finite hypothesis space H defined over the instance space X.
- Here the task is to learn some target concept c : X → {0,1}.
- As usual, we assume that the learner is given some sequence of training examples (<x1, d1>, <x2, d2>, <x3, d3>, . . <xm, dm>,) where xi is some instance from X and where di is the target value of xi.
- To simplify the discussion in this section, we assume the sequence of instances (x1 . . . xm) is held fixed, so that the training data D can be written simply as the sequence of target values D = (d1 . . . Dm)

# Brute-Force Bayes Concept Learning

- **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 Bayes Concept Learning

- **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 Bayes Concept Learning

- **BRUTE-FORCE MAP LEARNING algorithm**
- Brute Force MAP learning algorithm must specify values for P(h) and P(D|h).
- P(h) and P(D|h) must be chosen to be consistent with the assumptions:

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

# Brute-Force Bayes Concept Learning

- 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 Bayes Concept Learning

- What choice shall we make for **P(D|h)?**
- **P(D|h) is the probability of observing** the target values D = **<d1 . . .dm> for the fixed set of instances <X1 . . . Xm>.**
- Since we assume noise-free training data, the probability of observing classification **di given h is just 1 if di = h(xi)** and 0 if **di != h(xi).**



- **Therefore,**

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

- In other words, the probability of data **D given hypothesis h is 1 if D is consistent**
- with **h, and 0 otherwise.**

# Brute-Force Bayes Concept Learning

- Given these choices for **P(h) and for P(D|h) we now have a fully-defined** problem for the above **BRUTE-FORCE MAP LEARNING agorithm.**
- **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 Bayes Concept Learning

- 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

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



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

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

# Brute-Force Bayes Concept Learning

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

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

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

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

# Brute-Force Bayes Concept Learning

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

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

# 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.

# MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

- 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.

# MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

- In order to find the maximum likelihood hypothesis, we start with our earlier 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

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

# MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

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

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



- We now apply a transformation that is common in maximum likelihood calculations
- Rather than maximizing the above complicated expression we shall choose to maximize its (less complicated) logarithm.
- This is justified because $\ln(p)$ is a monotonic function of ***p. Therefore maximizing In p also maximizes p.***

# MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

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

- 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.

# MINIMUM DESCRIPTION LENGTH PRINCIPLE

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

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



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

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

- or alternatively, minimizing the negative of this quantity

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

# MINIMUM DESCRIPTION LENGTH PRINCIPLE

- To explain this, let us introduce a basic result from information theory: Consider the problem of designing a code to transmit messages drawn at random, where the probability of encountering message **i is pi.**
- **We are interested here in the most compact code; that is, we are interested in** the code that minimizes the expected number of bits we must transmit in order to encode a message drawn at random.
- Clearly, 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 **$-\log_2 p_i$** bits to encode message i **.**
- **We will** refer to 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

- Let us interpret $h_{MAP}$ Equation in light of the above result from coding theory.
- **$-\log_2 P(h)$** is the description length of h under the optimal encoding for the hypothesis space H.
- In other words, this is the size of the description of hypothesis h using this optimal representation.



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

# MINIMUM DESCRIPTION LENGTH PRINCIPLE

- Therefore we can rewrite Equation 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.

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



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

# MINIMUM DESCRIPTION LENGTH PRINCIPLE

- **Minimum Description Length principle:**
- **Choose $h_{MDL}$ *where***

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



- 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 $h_{MDL} = h_{MAP}$

# NAIVE BAYES CLASSIFIER

- The Bayesian approach to classifying the new instance is to assign the most probable target value, $v_{MAP}$ given the attribute values **<a1, a2 . . .an> that describe** the instance.

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

- We can use Bayes theorem to rewrite this expression as



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

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

**Naive Bayes classifier:**

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

# NAIVE BAYES CLASSIFIER

- One highly practical Bayesian learning method is the naive Bayes learner, often called the ***naive Bayes classijier.***
- ***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 ***<al, a2…an>.***
- ***The learner is asked to predict the target*** value, or classification, for this new instance.

# NAIVE BAYES CLASSIFIER - An Illustrative Example

| 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 CLASSIFIER - An Illustrative Example

- Here there are 14 training examples of the target concept PlayTennis, where each day is described by the attributes Outlook, Temperature, Humidity, and Wind.
- Here we use the naive Bayes classifier and the training data from this table to classify the following novel instance:



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

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

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

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

# NAIVE BAYES CLASSIFIER - An Illustrative Example

| Outlook | P | 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 | P | N |
|---|---|---|
| high | 3/9 | 4/5 |
| normal | 6/9 | 1/5 |
| Windy | | |
| true | 3/9 | 3/5 |
| false | 6/9 | 2/5 |

# NAIVE BAYES CLASSIFIER - An Illustrative Example

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

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

$$P(Wind = strong|PlayTennis = yes) = 3/9 = .33$$

$$P(Wind = strong|PlayTennis = no) = 3/5 = .60$$



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

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

- Thus, the naive Bayes classifier assigns the target value ***PlayTennis = no*** *to this* new instance

# NAIVE BAYES CLASSIFIER - An Illustrative Example

- Estimate conditional probabilities of each attributes {color, legs, height, smelly} for the 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 |

| 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

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



| 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 |

New Instance

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

| 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 |



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

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

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

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

$$\boldsymbol{p(H|New\ Instance) > p(M|New\ Instance)}$$

***Hence the new instance belongs to Speces H***

# NAIVE BAYES CLASSIFIER - AN EXAMPLE

| 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 |

# 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 |

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

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

| 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 |



$New\ Instance = (Red, SUV, Domestic)$ **No**

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

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

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

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

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

CSE, HIT, Nidasoshi

# NAIVE BAYES CLASSIFIER - AN EXAMPLE: LEARNING TO CLASSIFY TEXT

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 $P(X_B|Y_A) = 0.35$.

# NAIVE BAYES CLASSIFIER - AN EXAMPLE: LEARNING TO CLASSIFY TEXT

The above question can be solved by computing $P(Y_B|X_B)$, which is the conditional probability that team B wins the next match it hosts. Using the Bayes theorem, we obtain:

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



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

$$= \frac{0.75 \times 0.35}{(0.75 \times 0.35 + 0.35 \times 0.65)} = 0.5357$$

# NAIVE BAYES CLASSIFIER EXAMPLE – 4

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 $P(X_B|Y_A) = 0.35$.

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

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

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

$$= 0.5357$$

# NAIVE BAYES CLASSIFIER EXAMPLE – 4

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 $P(Y_B) = 1 - P(Y_A) = 0.35$

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 $P(X_B|Y_A) = 0.35$.

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

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

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

$$= 0.4642$$

CSE, HIT, Nidasoshi

# BAYESIAN BELIEF NETWORKS

- Bayesian networks (BN) are probabilistic graphical models that are based on directed acyclic graphs.
- They provide a tool to deal with two problems: uncertainty and complexity.
- Hence, they provide a compact representation of joint probability distributions using a combination of graph theory and probability theory.
- The graph structure specifies statistical dependencies among the variables and the local probabilistic models specify how these variables are combined.

# BAYESIAN BELIEF NETWORKS

- The naive Bayes classifier makes significant use of the assumption that the values of the attributes **a1 . . .an** are conditionally independent given the target value v.
- This assumption dramatically reduces the complexity of learning the target function
- A Bayesian belief network describes the probability distribution governing a set of variables by specifying a set of conditional independence assumptions along with a set of conditional probabilities
- Bayesian belief networks allow stating conditional independence assumptions that apply to subsets of the variables

# BAYESIAN BELIEF NETWORKS

- Consider an arbitrary set of random variables Y1 . . . Yn , where each variable Yi can take on the set of possible values V(Yi).
- The joint space of the set of variables Y to be the cross product V(Y1) x V(Y2) x. . . V(Yn).
- In other words, each item in the joint space corresponds to one of the possible assignments of values to the tuple of variables (Y1 . . . Yn). The probability distribution over this joint' space is called the joint probability distribution.
- The joint probability distribution specifies the probability for each of the possible variable bindings for the tuple (Y1 . . . Yn).
- A Bayesian belief network describes the joint probability distribution for a set of variables.

# BAYESIAN BELIEF NETWORKS

- 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 $z_k \in V(Z)$.

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

- 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

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

# BAYESIAN BELIEF NETWORKS - REPRESENTATION

- The joint probability for any desired assignment of values **(y1,** . . . , **yn)** to the tuple of network variables **(Y1** . . . **Yn)** can be computed by the formula

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



- 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.**

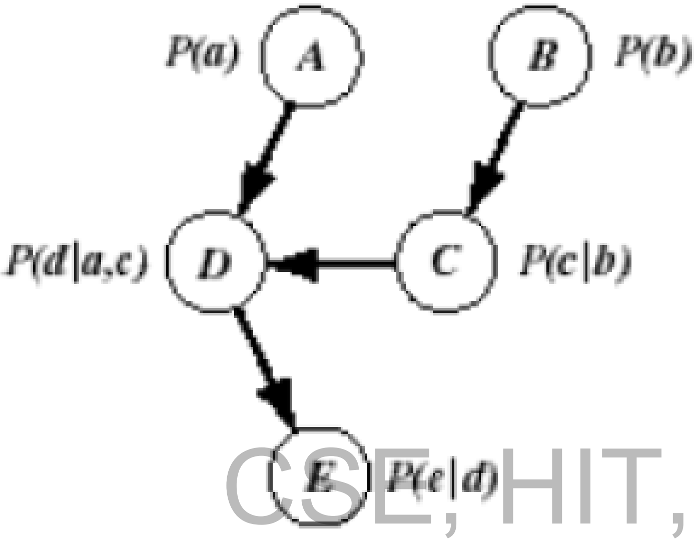



Figure 2: $P(a,b,c,d,e) = P(a)P(b)P(c|b)P(d|a,c)P(e|d)$

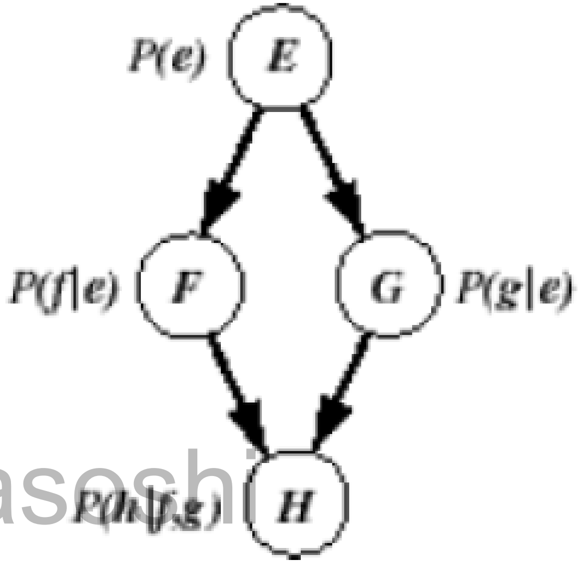



Figure 4: $P(e,f,g,h) = P(e)P(f|e)P(g|e)P(h|f,g)$

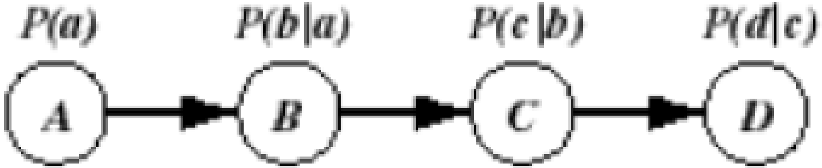
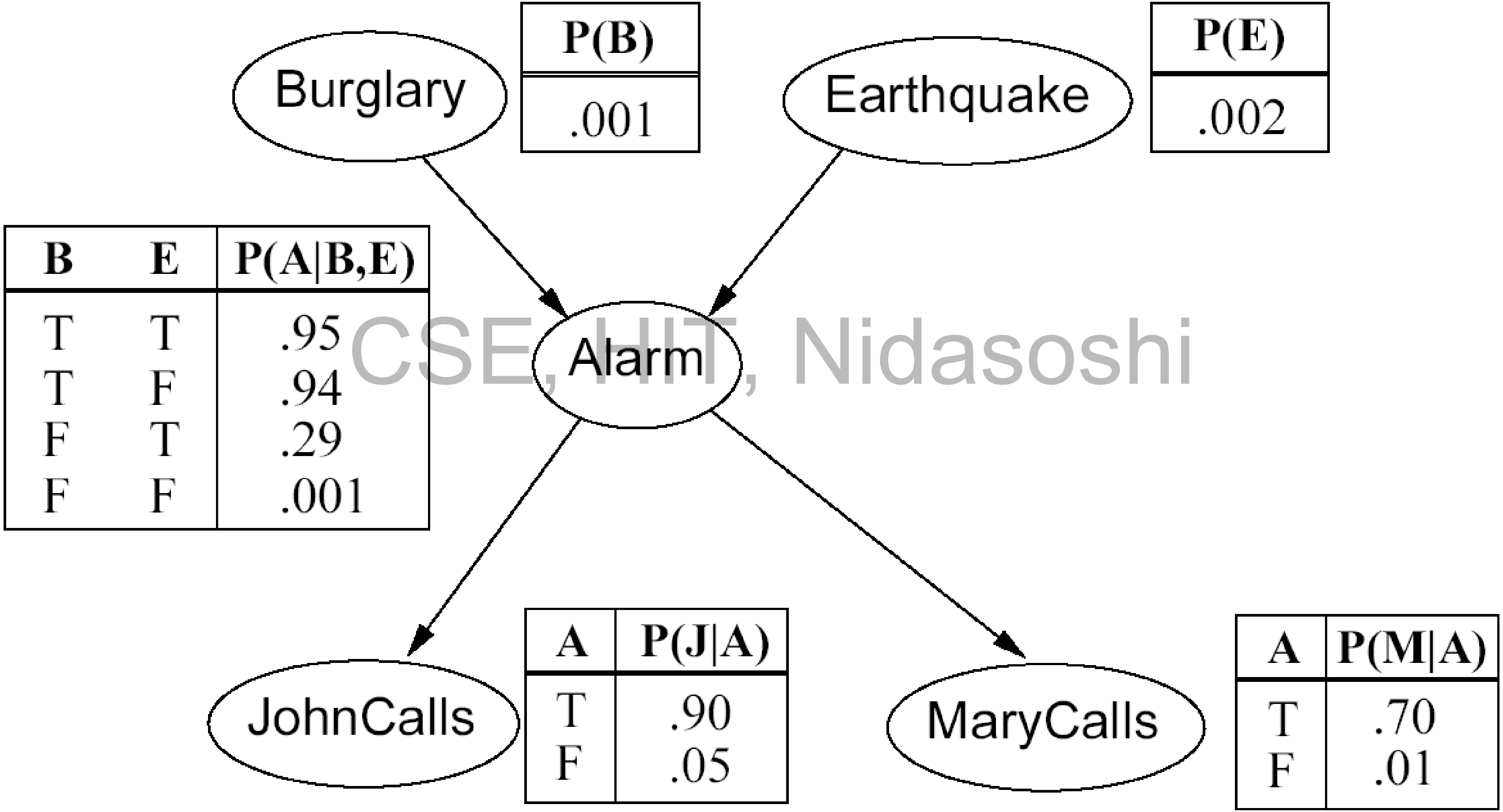



Figure 3: $P(a,b,c,d) = P(a)P(b|a)P(c|b)P(d|c)$

# BAYESIAN BELIEF NETWORKS - EXAMPLE

- 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




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

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

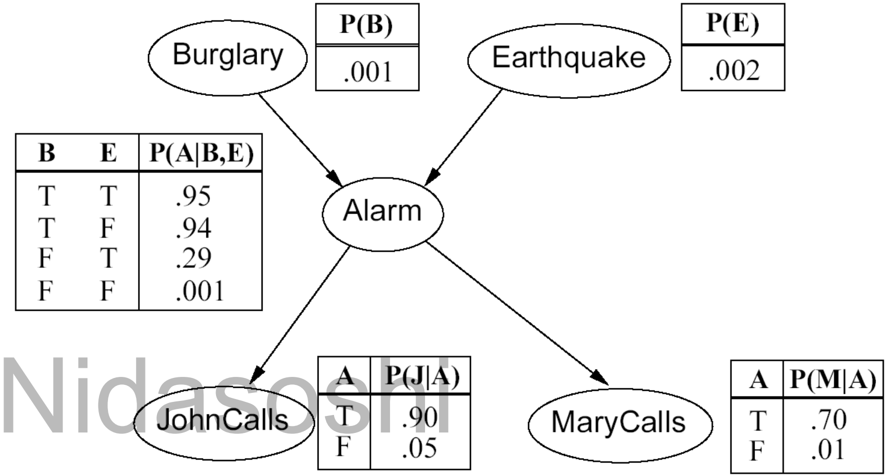
| B | E | P(A\|B,E) |
|---|---|---|
| T | T | .95 |
| T | F | .94 |
| F | T | .29 |
| F | F | .001 |

| A | P(J\|A) |
|---|---|
| T | .90 |
| F | .05 |

| A | P(M\|A) |
|---|---|
| T | .70 |
| F | .01 |

# BAYESIAN BELIEF NETWORKS - EXAMPLE

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:**

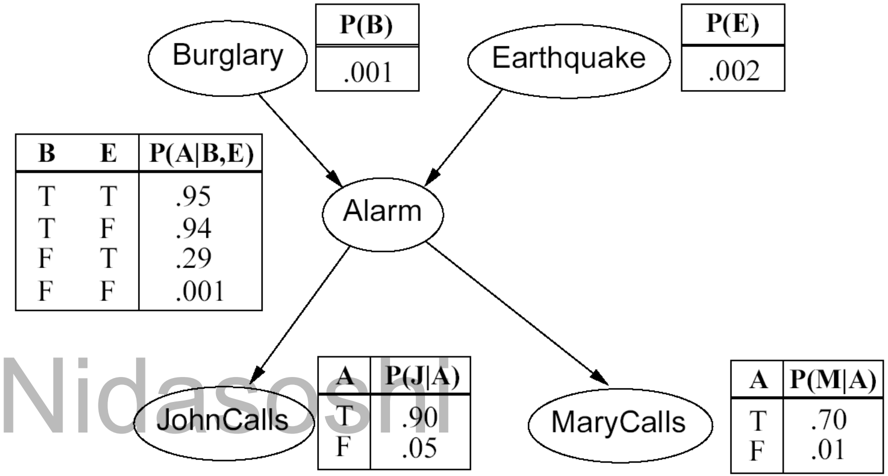
$$P(j \wedge m \wedge a \wedge \neg b \wedge \neg e) = P(j \mid a)\, P(m \mid a)\, P(a \mid \neg b, \neg e)\, P(\neg b)\, P(\neg e)$$

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

$$= 0.00062$$

# BAYESIAN BELIEF NETWORKS - EXAMPLE

2. What is the probability that John call?




Solution:



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

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

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

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

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

CSE, HIT, Nidasoshi

# BAYESIAN BELIEF NETWORKS - EXAMPLE

- Suppose, we are given for the evidence variables $E_1,\ldots,E_m$, their values $e_1,\ldots,e_m$, and we want to predict whether the query variable $X$ has the value $x$ or not.



- For this we compute and compare the following:

$$P(x \mid e_1,\ldots,e_m) = \frac{P(x, e_1,\ldots,e_m)}{P(e_1,\ldots,e_m)} = \alpha P(x, e_1,\ldots,e_m)$$

$$P(\neg x \mid e_1,\ldots,e_m) = \frac{P(\neg x, e_1,\ldots,e_m)}{P(e_1,\ldots,e_m)} = \alpha P(\neg x, e_1,\ldots,e_m)$$

$$\propto$$

$$= \frac{1}{(P(x,e_1,\ldots em) + P(\neg x,e_1,\ldots em))}$$

# BAYESIAN BELIEF NETWORKS - EXAMPLE

3. What is the probability that there is a burglary given that John and Merry calls?

$$P(b \mid j, m)$$
$$= \alpha P(b, j, m)$$
$$= \alpha \sum_a \sum_e P(b, j, m, a, e)$$
$$= \alpha \big(P(b, j, m, a, e) + P(b, j, m, \neg a, e) + P(b, j, m, a, \neg e) + P(b, j, m, \neg a, \neg e)\big)$$



$$P(\neg b \mid j, m)$$
$$= \alpha P(\neg b, j, m)$$
$$= \alpha \sum_a \sum_e P(\neg b, j, m, a, e)$$
$$= \alpha \big(P(\neg b, j, m, a, e) + P(\neg b, j, m, \neg a, e) + P(\neg b, j, m, a, \neg e) + P(\neg b, j, m, \neg a, \neg e)\big)$$

# BAYESIAN BELIEF NETWORKS - EXAMPLE

3. What is the probability that there is a burglary given that John and Merry calls?

$$P(b \mid j,m) = \alpha\, P(b) \textstyle\sum_a P(j|a)P(m|a) \sum_e P(a|b,e)\, P(e)$$

$$= \alpha\, P(b) \textstyle\sum_a P(j|a)\, P(m|a)\, \{P(a|b,e)P(e) + P(a|b,\neg e)P(\neg e)\}$$



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

$$+ P(j|\neg a)P(m|\neg a)\, \{P(\neg a|b,e)P(e) + P(\neg a|b,\neg e)P(\neg e)\}\,]$$

$$= \alpha * .001*(.9*.7*(.95*.002 + .94*.998) +.05*.01*(.05*.002 + .71*.998)\,)$$

$$= \alpha * .00059$$

# BAYESIAN BELIEF NETWORKS - EXAMPLE

3. What is the probability that there is a burglary given that John and Merry calls?

$$P(\neg b \mid j,m) = \alpha\, P(\neg b) \textstyle\sum_a P(j|a)P(m|a)\sum_e P(a|\neg b,e)P(e)$$

$$= \alpha\, P(\neg b) \textstyle\sum_a P(j|a)P(m|a) \{ P(a|\neg b,e)P(e) + P(a|\neg b,\neg e)P(\neg e)\}$$



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

$$+ P(j|\neg a)P(m|\neg a) \{ P(\neg a|\neg b,e)P(e) + P(\neg a|\neg b,\neg e)P(\neg e) \}]$$

$$= \alpha * .999*(.9*.7*(.29*.002 + .001*.998) +.05*.01*(.71*.002 + .999*.998) )$$

$$= \alpha * .0015$$

# BAYESIAN BELIEF NETWORKS - EXAMPLE

3. What is the probability that there is a burglary given that John and Merry calls?

$$\alpha = 1/(.00059 + .0015)$$

$$= 478.5$$



$$P(b \mid j,m) = 478.5 * .00059$$

$$= 0.28$$

$$P(\neg b \mid j,m) = 478.5 * .0015$$

$$= 0.72$$

CSE, HIT, Nidasoshi

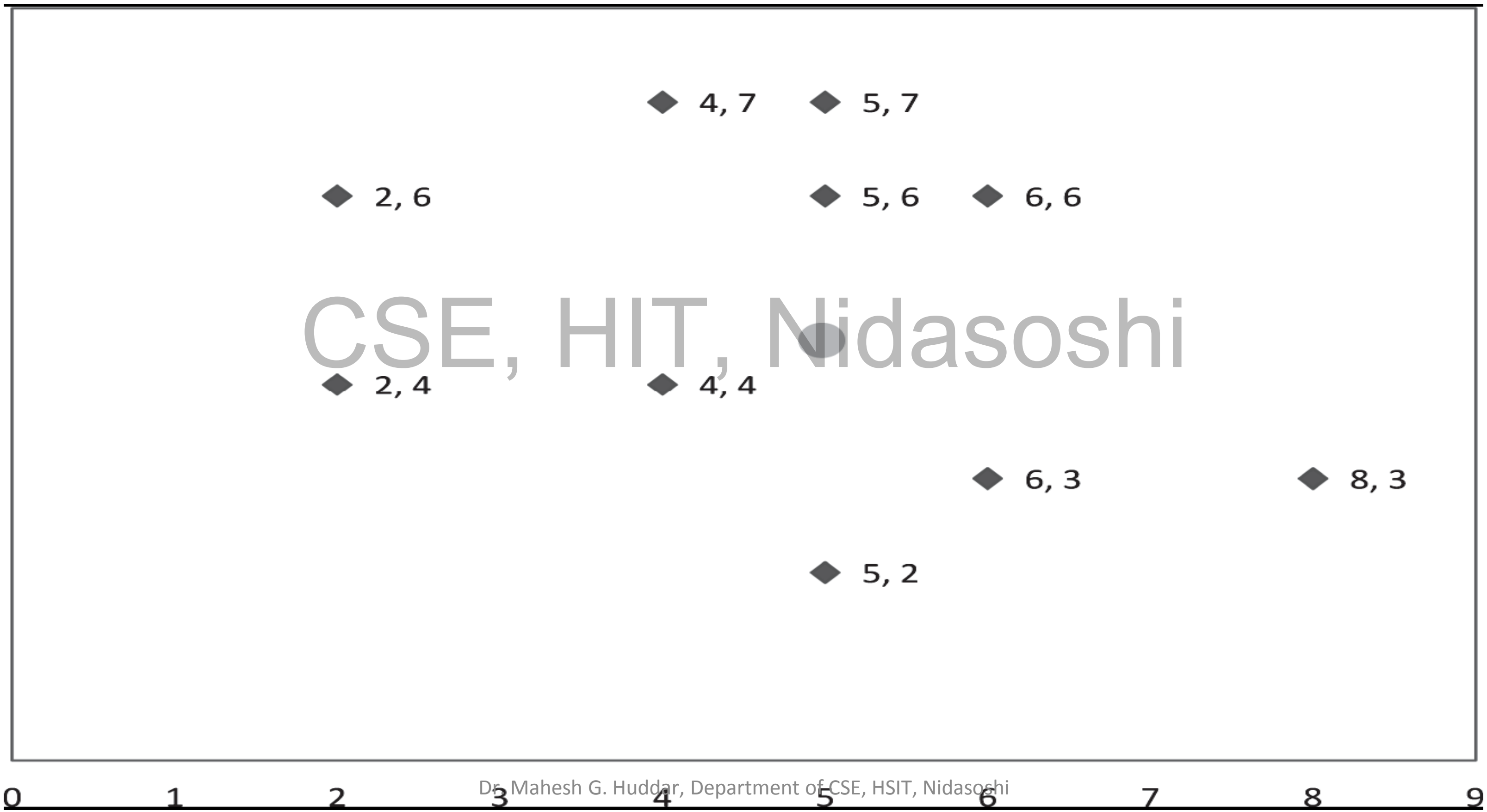
# K-Means Algorithm for Clustering

- 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 can use the Euclidean distance, Manhattan distance, Hamming distance, Cosine distance as measurement.

# K-Means Algorithm for Clustering



# 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).

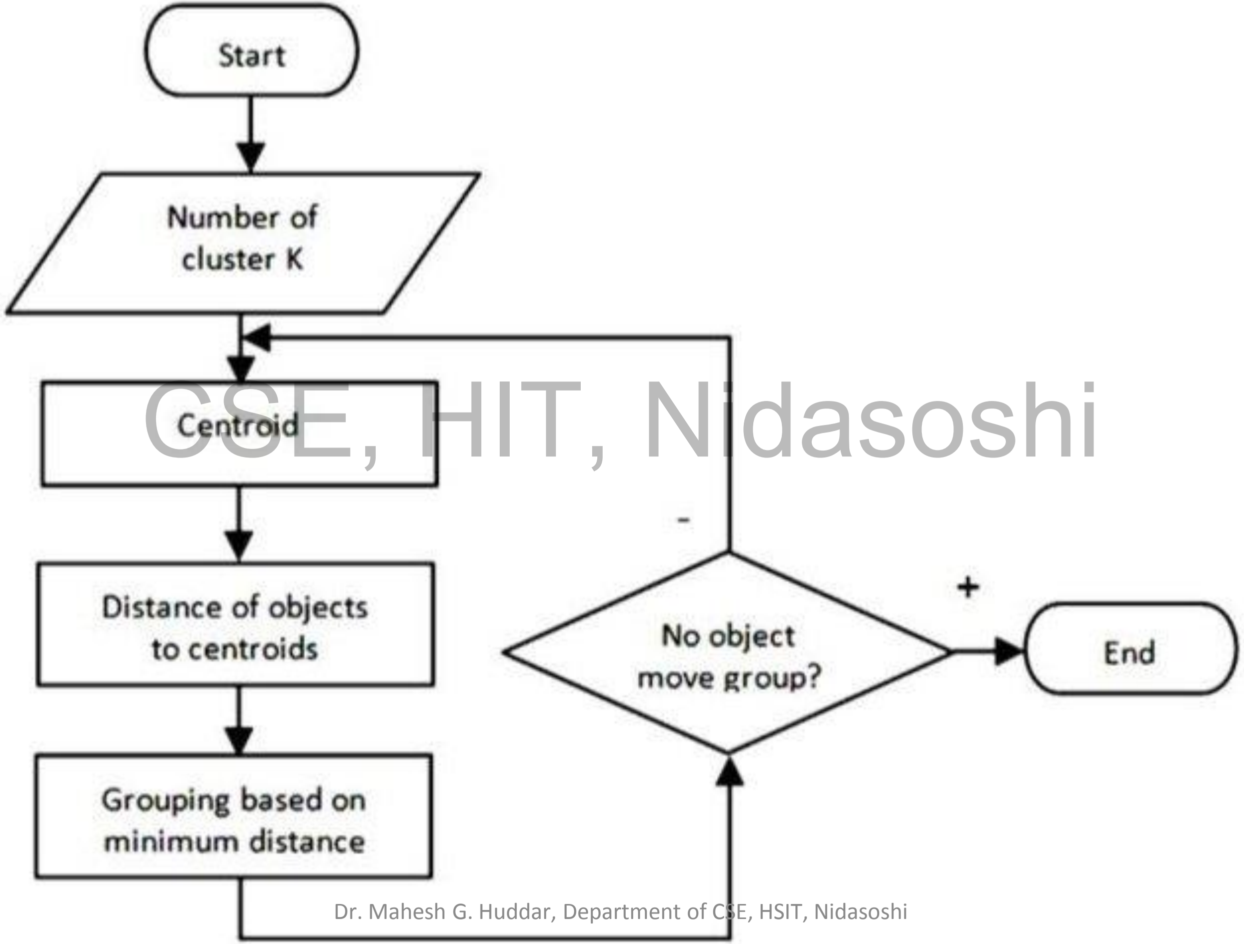


2. Repeat till cluster centers stabilize:

   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.



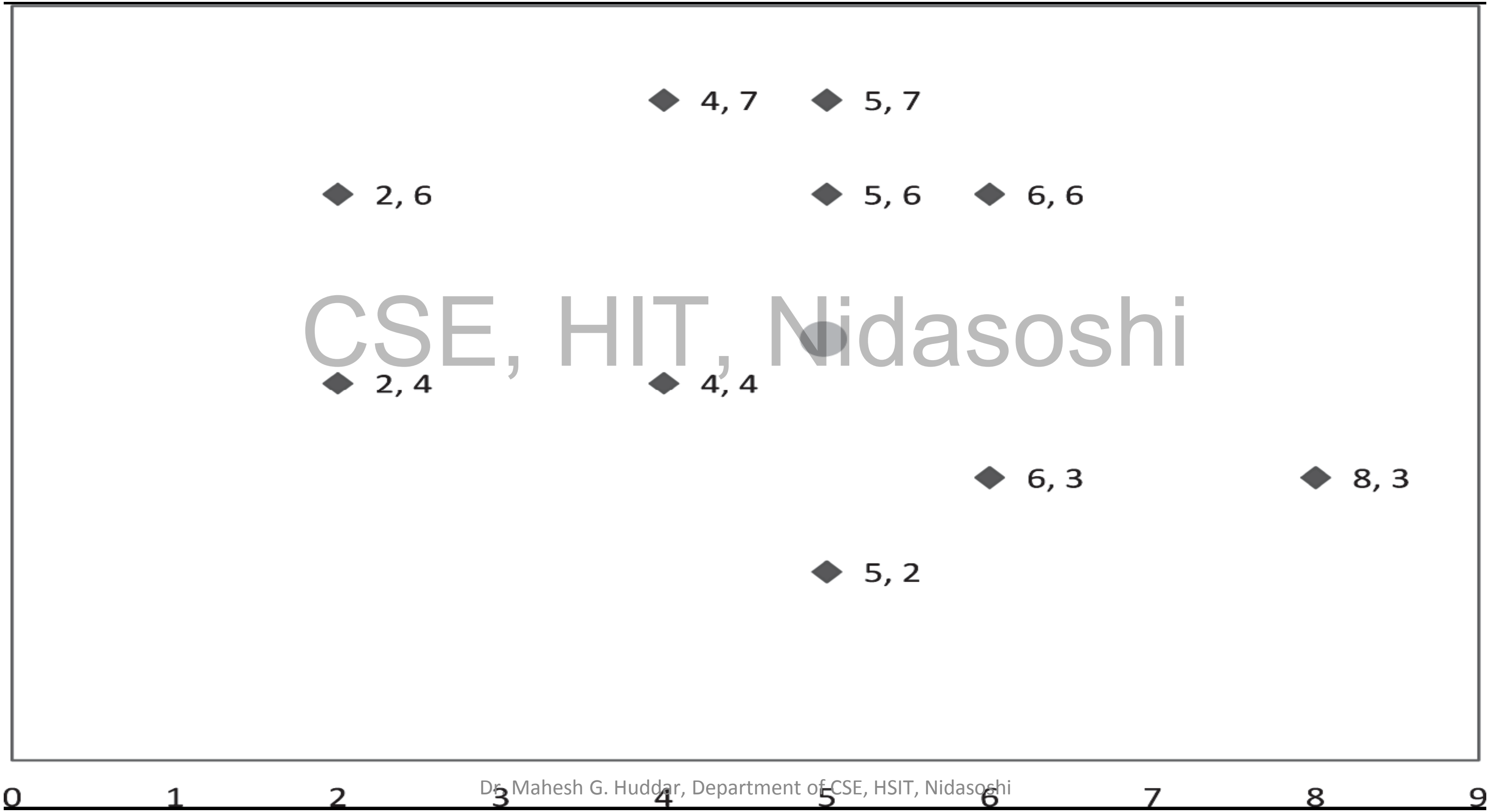
**Disadvantages of K-Means Algorithm**

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).

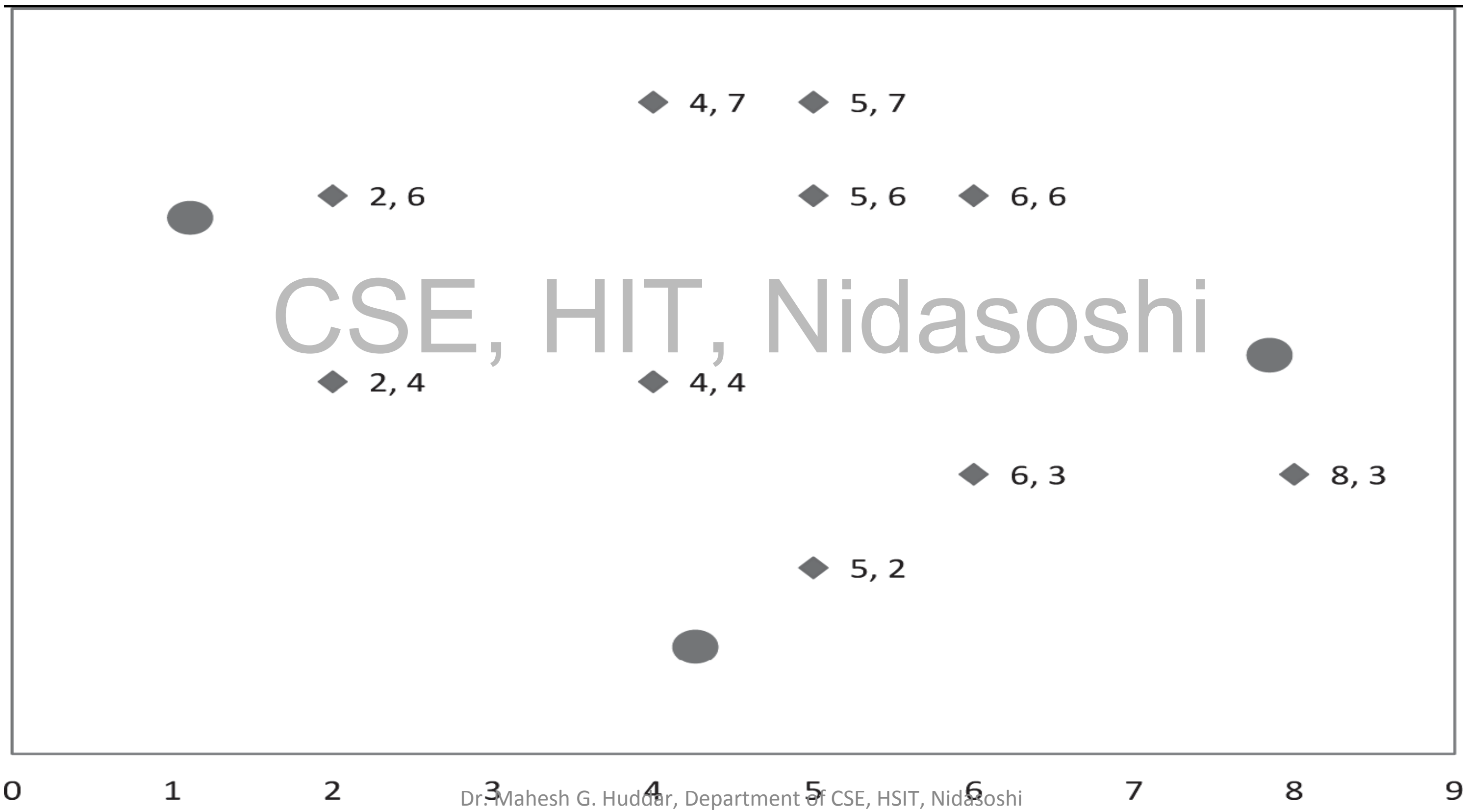
# Clustering Exercise

| X | Y |
|---|---|
| 2 | 4 |
| 2 | 6 |
| 5 | 6 |
| 4 | 7 |
| 8 | 3 |
| 6 | 6 |
| 5 | 2 |
| 5 | 7 |
| 6 | 3 |
| 4 | 4 |

# Clustering Exercise



# Clustering Exercise



# Clustering Exercise

**Iteration - 1**

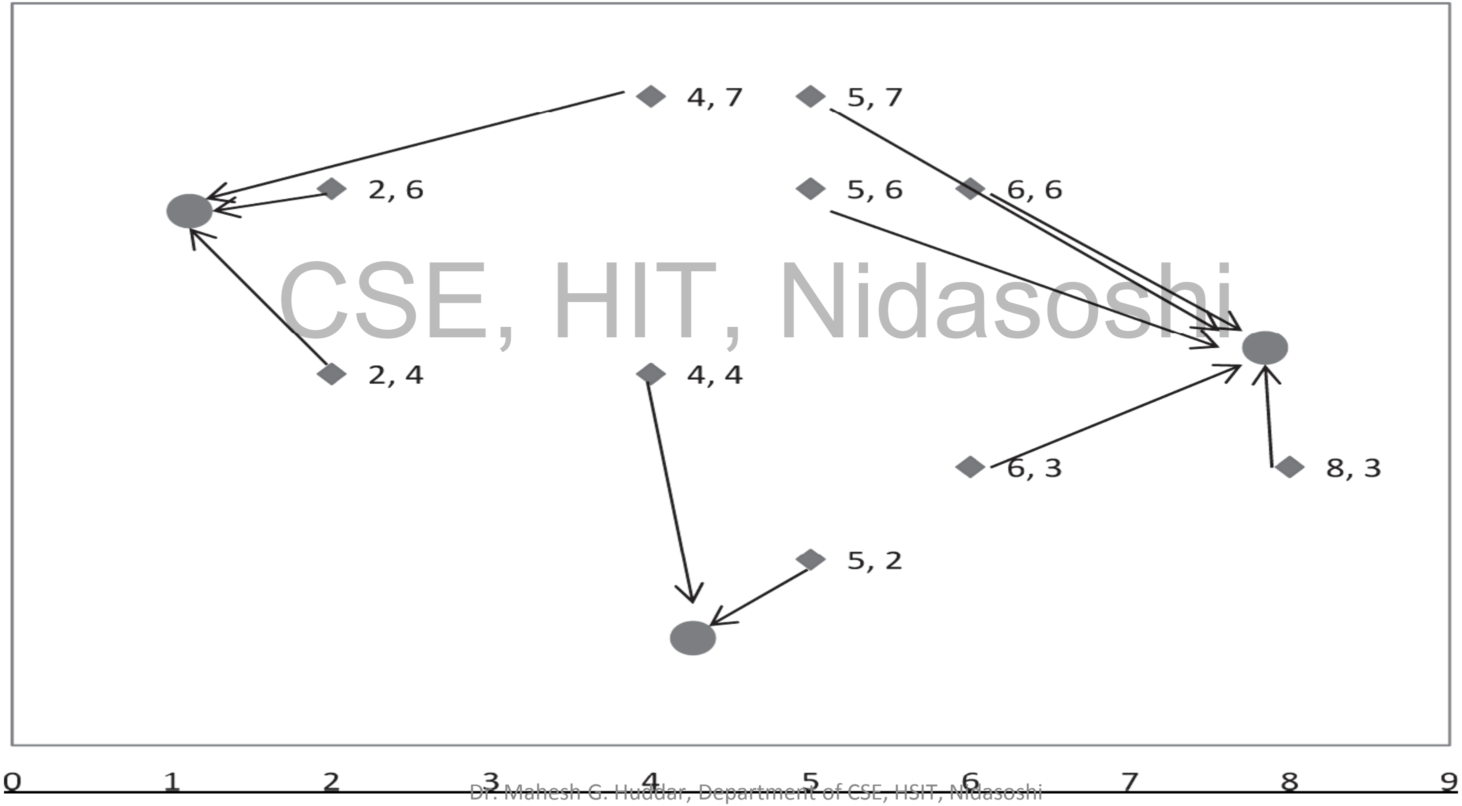
C1 - Seed Point1 – (1, 5)
C2 - Seed Point2 – (4, 1)
C3 - Seed Point3 – ( 8, 4)

$$D = \sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2)}$$

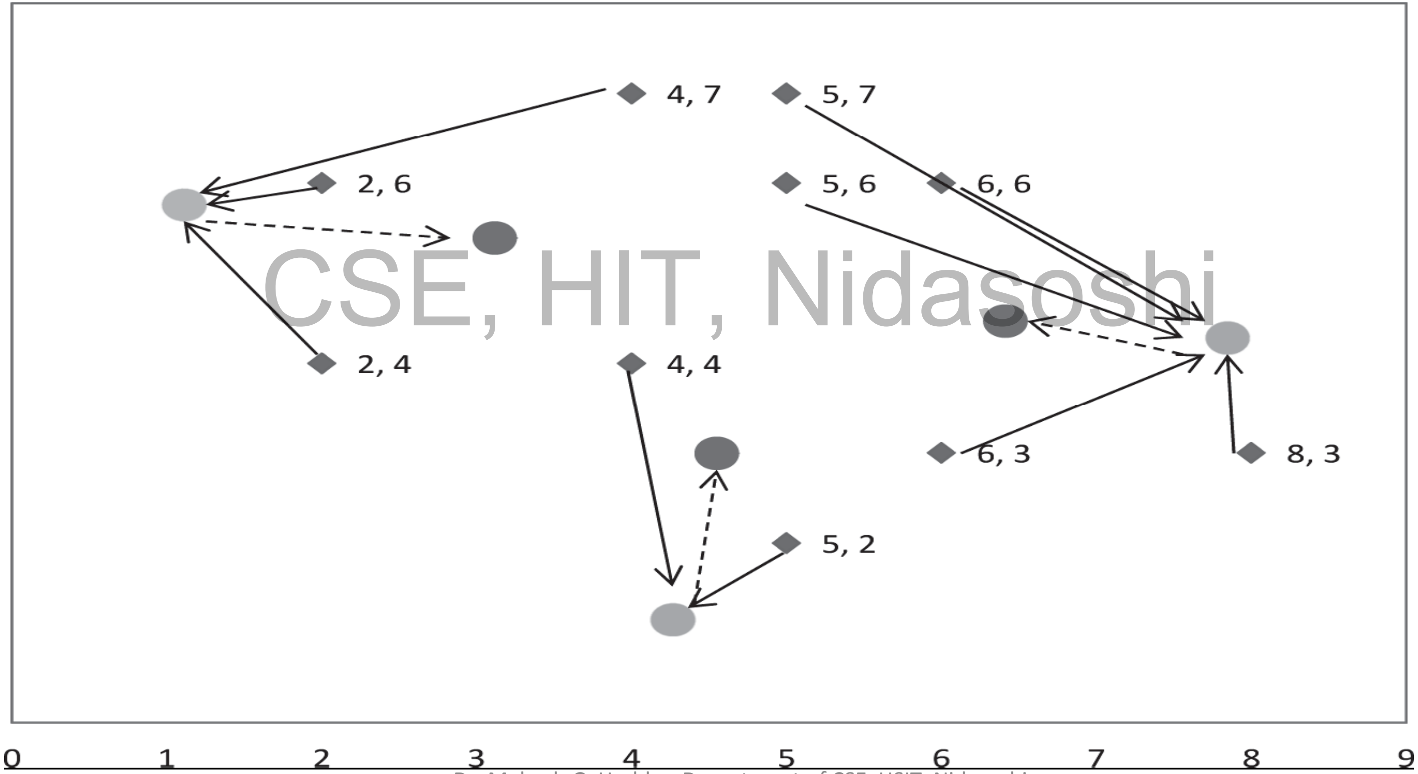
C1 – Centroid – (2.66, 5.66)
C2 – Centroid – (4.5, 3)
C3 – Centroid – ( 6, 5)

| | | Distance to | | | Cluster Number |
|---|---|---|---|---|---|
| X | Y | (1, 5) | (4, 1) | (8, 4) | |
| 2 | 4 | 1.41 | 3.61 | 6.00 | C1 |
| 2 | 6 | 1.41 | 5.39 | 6.32 | C1 |
| 5 | 6 | 4.12 | 5.10 | 3.61 | C3 |
| 4 | 7 | 3.61 | 6.00 | 5.00 | C1 |
| 8 | 3 | 7.28 | 4.47 | 1.00 | C3 |
| 6 | 6 | 5.10 | 5.39 | 2.83 | C3 |
| 5 | 2 | 5.00 | 1.41 | 3.61 | C2 |
| 5 | 7 | 4.47 | 6.08 | 4.24 | C3 |
| 6 | 3 | 5.39 | 2.83 | 2.24 | C3 |
| 4 | 4 | 3.16 | 3.00 | 4.00 | C2 |

# K-Means Algorithm for Clustering



# K-Means Algorithm for Clustering



# Clustering Exercise

**<u>Iteration - 2</u>**

C1 – Centroid – (2.66, 5.66)
C2 – Centroid – (4.5, 3)
C3 – Centroid – ( 6, 5)

C1 – Centroid – (2.66, 5.66)
C2 – Centroid – (5, 3)
C3 – Centroid – ( 6, 5.5)

| | | Distance to | | | Cluster Number |
|---|---|---|---|---|---|
| X | Y | (2.66, 5.66) | (4.5, 3) | (6, 5) | |
| 2 | 4 | 1.79 | 2.69 | 4.12 | C1 |
| 2 | 6 | 0.74 | 3.91 | 4.12 | C1 |
| 5 | 6 | 2.36 | 3.04 | 1.41 | C3 |
| 4 | 7 | 1.90 | 4.03 | 2.83 | C1 |
| 8 | 3 | 5.97 | 3.5 | 2.83 | C3 |
| 6 | 6 | 3.36 | 3.35 | 1 | C3 |
| 5 | 2 | 4.34 | 1.12 | 3.16 | C2 |
| 5 | 7 | 2.70 | 4.03 | 2.24 | C3 |
| 6 | 3 | 4.27 | 1.5 | 2 | C2 |
| 4 | 4 | 2.13 | 1.12 | 2.24 | C2 |

# Clustering Exercise

**Iteration - 3**

C1 – Centroid – (2.66, 5.66)
C2 – Centroid – (5, 3)
C3 – Centroid – ( 6, 5.5)

| | | Distance to | | | Cluster |
|---|---|---|---|---|---|
| **X** | **Y** | **(2.66, 5.66)** | **(5, 3)** | **(6, 5.5)** | **Number** |
| 2 | 4 | 1.79 | 3.16 | 4.27 | C1 |
| 2 | 6 | 0.74 | 4.24 | 4.03 | C1 |
| 5 | 6 | 2.36 | 3.00 | 1.12 | C3 |
| 4 | 7 | 1.90 | 4.12 | 2.50 | C1 |
| 8 | 3 | 5.97 | 3.00 | 3.20 | C2 |
| 6 | 6 | 3.36 | 3.16 | 0.50 | C3 |
| 5 | 2 | 4.34 | 1.00 | 3.64 | C2 |
| 5 | 7 | 2.70 | 4.00 | 1.80 | C3 |
| 6 | 3 | 4.27 | 1.00 | 2.50 | C2 |
| 4 | 4 | 2.13 | 1.41 | 2.50 | C2 |

C1 – Centroid – (2.66, 5.66)
C2 – Centroid – (5.75, 3)
C3 – Centroid – ( 5.33, 6.33)

# Clustering Exercise

**Iteration - 4**

C1 – Centroid – (2.66, 5.66)
C2 – Centroid – (5.75, 3)
C3 – Centroid – ( 5.33, 6.33)

C1 – Centroid – (2, 5)
C2 – Centroid – (5.75, 3)
C3 – Centroid – ( 5, 6.5)

| | | Distance to | | | Cluster Number |
|---|---|---|---|---|---|
| X | Y | (2.66, 5.66) | (5.75, 3) | (5.33, 6.33) | |
| 2 | 4 | 1.79 | 3.88 | 4.06 | C1 |
| 2 | 6 | 0.74 | 4.80 | 3.35 | C1 |
| 5 | 6 | 2.36 | 3.09 | 0.47 | C3 |
| 4 | 7 | 1.90 | 4.37 | 1.49 | C3 |
| 8 | 3 | 5.97 | 2.25 | 4.27 | C2 |
| 6 | 6 | 3.36 | 3.01 | 0.75 | C3 |
| 5 | 2 | 4.34 | 1.25 | 4.34 | C2 |
| 5 | 7 | 2.70 | 4.07 | 0.75 | C3 |
| 6 | 3 | 4.27 | 0.25 | 3.40 | C2 |
| 4 | 4 | 2.13 | 2.02 | 2.68 | C2 |

# Clustering Exercise

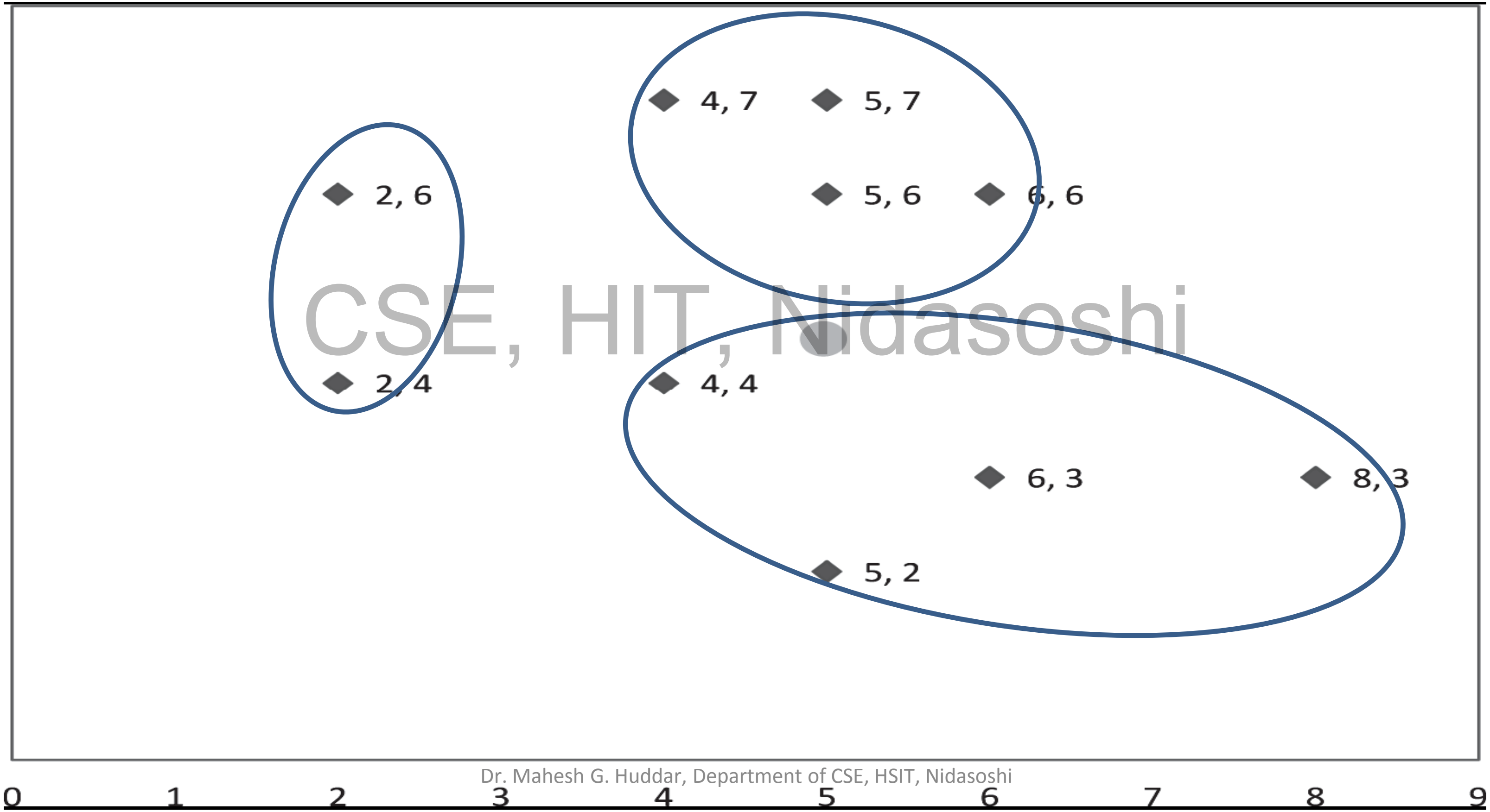
**Iteration - 5**

C1 – Centroid – (2, 5)
C2 – Centroid – (5.75, 3)
C3 – Centroid – ( 5, 6.5)

No movement of data Points Hence these are the final positions

| | | Distance to | | | Cluster Number |
|---|---|---|---|---|---|
| X | Y | (2, 5) | (5.75, 3) | (5, 6.5) | |
| 2 | 4 | 1.00 | 3.88 | 3.91 | C1 |
| 2 | 6 | 1.00 | 4.80 | 3.04 | C1 |
| 5 | 6 | 3.16 | 3.09 | 0.50 | C3 |
| 4 | 7 | 2.83 | 4.37 | 1.12 | C3 |
| 8 | 3 | 6.32 | 2.25 | 4.61 | C2 |
| 6 | 6 | 4.12 | 3.01 | 1.12 | C3 |
| 5 | 2 | 4.24 | 1.25 | 4.50 | C2 |
| 5 | 7 | 3.61 | 4.07 | 0.50 | C3 |
| 6 | 3 | 4.47 | 0.25 | 3.64 | C2 |
| 4 | 4 | 2.24 | 2.02 | 2.69 | C2 |

# Clustering Exercise



CSE, HIT, Nidasoshi

# MAXIMUM LIKELIHOOD AND LEAST-SQUARED ERROR HYPOTHESES

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

# EM ALGORITHM

- In the real-world applications of machine learning, it is very common that there are many relevant features available for learning but only a small subset of them are observable.
- So, for the variables which are sometimes observable and sometimes not, then we can use the instances when that variable is visible is observed for the purpose of learning and then predict its value in the instances when it is not observable.
- The ***Expectation-Maximization algorithm*** can be used for the latent variables (variables that are not directly observable and are actually inferred from the values of the other observed variables) in order to predict their values with the condition that the general form of probability distribution governing those latent variables is known to us.
- This algorithm is actually at the base of many unsupervised clustering algorithms in the field of machine learning.

# EM ALGORITHM

Let us understand the EM algorithm in detail.

- Initially, a set of initial values of the parameters are considered.
- A set of incomplete observed data is given to the system with the assumption that the observed data comes from a specific model.
- The next step is known as "Expectation" – step or *E-step*. In this step, we use the observed data in order to estimate or guess the values of the missing or incomplete data. It is basically used to update the variables.
- The next step is known as "Maximization"-step or *M-step*. In this step, we use the complete data generated in the preceding "Expectation" – step in order to update the values of the parameters. It is basically used to update the hypothesis.
- Now, in the fourth step, it is checked whether the values are converging or not, if yes, then stop otherwise repeat *step-2* and *step-3* i.e. "Expectation" – step and "Maximization" – step until the convergence occurs.

# EM ALGORITHM

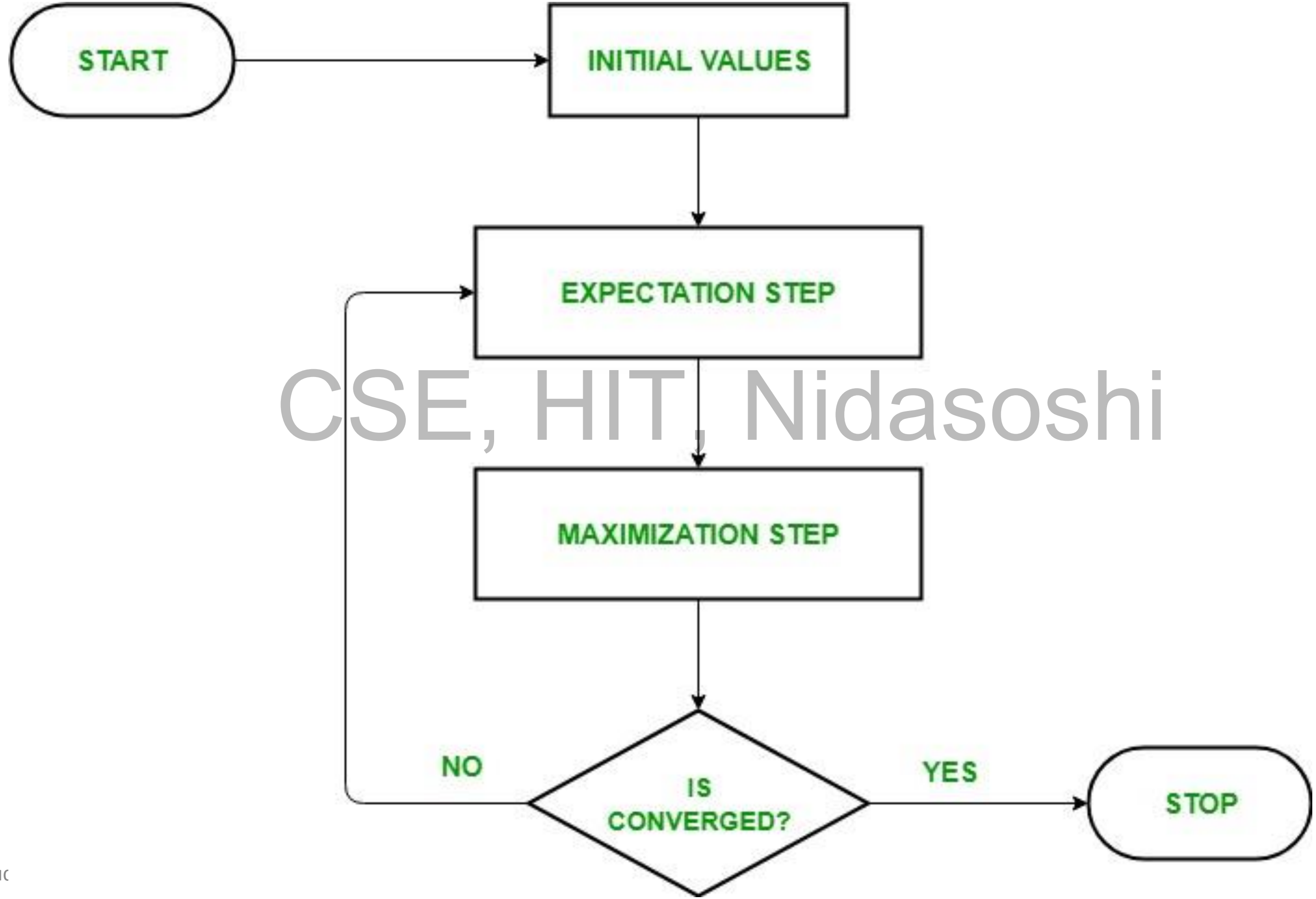
**Algorithm:**

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.

# 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.

# EM ALGORITHM

**Disadvantages of EM algorithm –**

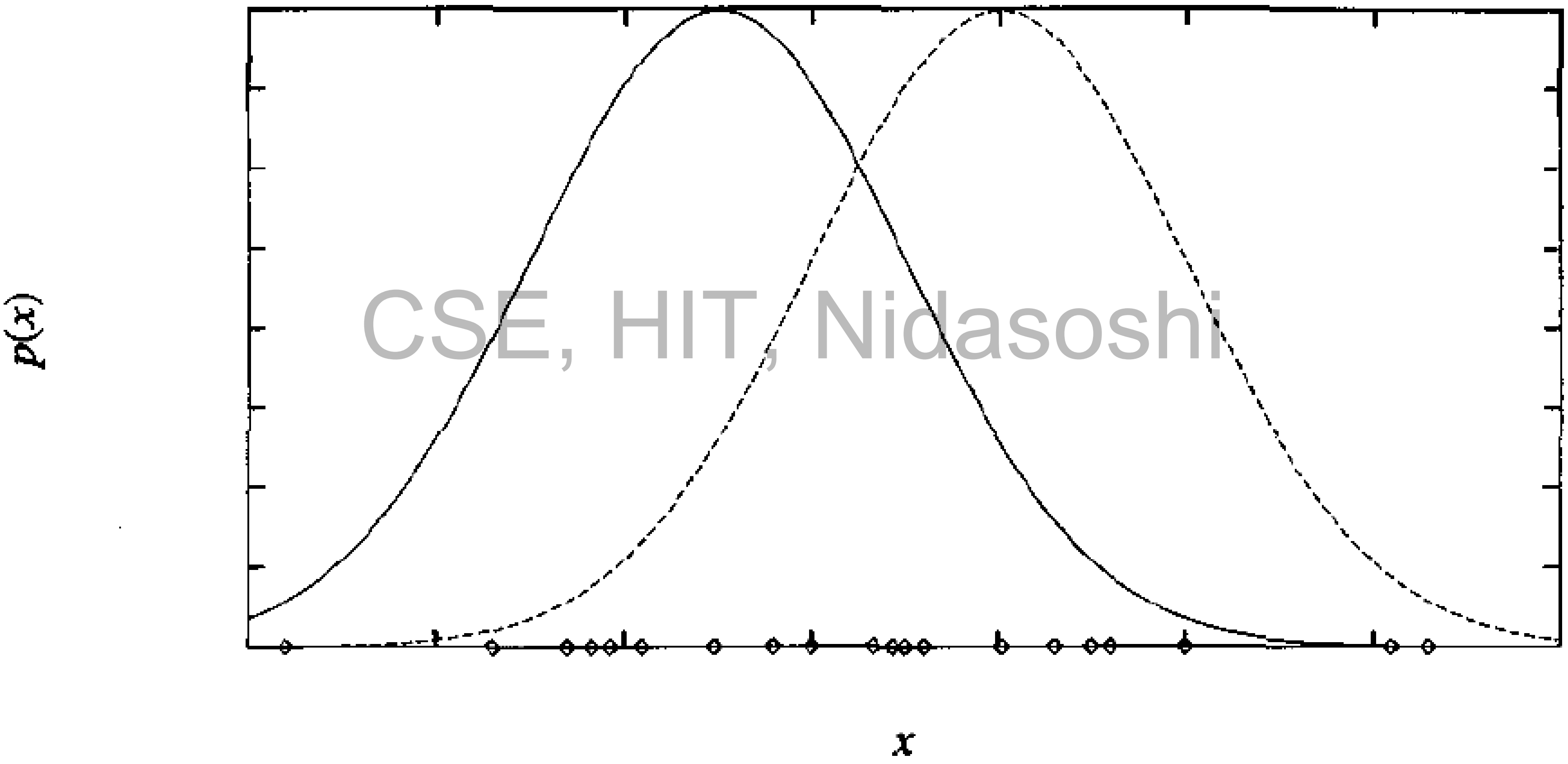
- It has slow convergence.
- It makes convergence to the local optima only.
- It requires both the probabilities, forward and backward (numerical optimization requires only forward probability).

CSE, HIT, Nidasoshi

# Estimating Means of k Gaussians

- The easiest way to introduce the EM algorithm is via an example.
- Consider a problem in which the data D is a set of instances generated by a probability distribution that is a mixture of k distinct Normal distributions.
- 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.

# Estimating Means of k Gaussians



# 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.

# Estimating Means of k Gaussians

- 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(Ylh'),** as we have discussed on several occasions already.
- Third, we introduce the expected value **E[ln P(Ylh')]** 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.

# Estimating Means of k Gaussians

- 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 lh')]** 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.

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

# 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 = ***{ < z i1 ,*** . . . , ***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 ***lnp(Y|h'),***
- Note the probability ***p(yi |h')*** of a single instance ***yi*** = ***(xi,zi1 ,*** . . .z***ik) 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 In ***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 In P(Y|h') over the probability distribution governing Y

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

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



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

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

# Derivation of the k Means Algorithm

- 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 $\mu_j'$ is weighted by E[zij].