MODEL- BASED CLUSTERING METHODS

- One disadvantage of hierarchical clustering algorithms, K-means algorithms and others is that they are largely heuristic and not based on formal models. Formal inference is not possible.

[Notes:- A heuristic technique is any approach to problem solving, learning or discovery that employs a practical method, not guaranteed to be optimal, perfect, logical or rational, but instead sufficient for reaching an immediate goad]

- Hence, model based clustering is an alternative.
- Model - based clustering methods attempts to optimize the fit betnuen the given data and some mathematical model.

- Such methods are often based on the assumption that the data are generated by a mixture of underlying probability distributions.

Before jumping to model based clustering methods, recall the probability distributions-

1) Bernoulli distribution
2) uniform distribution
3) Binomial dis.
4) Normal dist.
5) Poisson
6) Exponential

Quick recap of Normal distribution (required later).

The probability density of the normal distribution is →

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

where

μ → the mean of the distribution

σ → standard deviation.

I. EXPECTATION - MAXIMIZATION

- Each cluster can be represented mathematically by a probability distribution, where each individual distribution is typically referred to as component distribution.
- The entire data is a mixture of these distribution.
- These mixture-models are probabilistic-grounded way of doing soft clustering

hard clustering: elements either belong to the cluster or not.
soft clustering: strength of association b/w clusters and instances

- We can therefore cluster the data using a finite mixture density model of K probability distribution, where each distribution represents a cluster.
- Therefore, the problem is to estimate the parameters of the probability distributions so as to best fit the data.
  ⇒(E.g. if we have normal / gaussian distribution, we need to estimate the mean and covariance of each component distribution).

※ The EM algorithm is a popular iterative refinement algorithm that can be used for finding the parameter estimates.

※ It can be viewed as an extension of the K-means paradigm, which assigns an object to the cluster with which it is most similar, based on the cluster mean.

※ Instead of assigning each object to a dedicated cluster, EM assigns each object with a to a cluster according to a weight representing the probability of membership.

To understand this better,
suppose that we have n observations $(x_1, x_2 \ldots x_n)$ and for $k=2$, the gaussians are ~~xxxxx~~ unknown.
(i.e. unknown μ and σ^2. But we have source of each ~~p~~ observations, i.e. we know which point belongs to which gaussian.

• → gaussian 1
• → gaussian 2

In this case using mean & s.d. formula, we can always calculate the gaussian parameters.

$$\mu_b = \frac{x_1 + x_2 + \cdots + x_{n_b}}{n_b} \qquad \sigma_b^2 = \frac{(x_1-\mu_b)^2 + \cdots + (x_n-\mu_b)^2}{n_b}$$

- what if we dont know the sources! what if we just have the bunch of few data sources, and we dont know which ~~source~~ point came from which source. Can we fit the gaussian now?.
- This exact same situation is tackled by mixture models, where we know that the data points comes from k-gaussians, but dont know which point came from which.

Algorithm:

(1) Make an initial guess of the parameter vector: This involves randomly selecting k objects to represent the cluster means, as well as making guesses for additional parameters.

(2) Iteratively refine the parameters based on two steps -

(a) <u>Expectation step</u>: Assign to each object x_i to the cluster C_k with the probability -

$$P(x_i \in C_k) = p(C_k|x_i)$$
$$= \frac{p(C_k)\, p(x_i|C_k)}{p(x_i)}$$

[Bayes Th^m]

$\left(P(x_i) = P(x_i|C_1) + P(x_i|C_2) + \cdots + P(x_i|C_k) \right)$

where
$$\boxed{P(x_i|C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}}\, e^{-\frac{(x_i-\mu_k)^2}{2\sigma_k^2}}}$$
normal distribution.

- Hence this step calculates the prob. of cluster membership of object x_i, for each of the clusters.

(b) Maximization ~~problem~~ step:
using the prob. estimates from the above to re-estimate (or to refine the model parameters.

$$\mu_K = \frac{1}{n} \sum_{i=1}^{n} \frac{x_i P(x_i \in C_K)}{\sum_j P(x_i \in C_j)}$$

This step is the maximization of the likelihood of the distributions given

⇒ computational complexity → linear to
d (no. of input features)
n (no. of objects)
t (no. of iterations)

(AutoClass) → book.

Scanned by CamScanner

CONCEPTUAL CLUSTERING

- It is a form of clustering in ML that, given a set of unlabeled object, produces a classification scheme over the objects.
- Unlike conventional clus.

  ↳ a step further ⇒ also finds characteristic descriptions for each group.

  (Each group represents a concept / class).

- So, conceptual clus ⇒ two step process -

  step - ① clustering

  step - ② characterisation.

⇒ Generally we use probabilistic descriptions to represent each derived class.

⇒ Method used : COBWEB.

it is an incremental conceptual clustering.

Input objects ⇒ described as ⇒ attribute value pair.

Output ⇒ classification tree.

How it is different from decision tree?

- Each node in a classification tree refers to a concept and contains a probabilistic description of that concept.
- The probabilistic description includes the prob. of <u>concept</u> and <u>conditional prob.</u>

  of form $P(A_i = v_{ij} \mid C_k)$

  $A_i = v_{ij}$ → attribute value pairs

  i.e. for a given attribute A_i, it takes v_{ij} th value

  ⇙

Probab. that a certain class C_k has v_{ij} as the value of attribute A_i.

4.

- This is unlike decision trees, where each ~~e~~ leaf node is label ~~rather~~ rather than each node.
(and ~~a~~ decision uses other parameters for branching info. gain & entropy).

⇒ Now cobweb uses an evaluation measure called "category utility" to guide cons. of tree.

$$CU = \frac{\sum_{k=1}^{n} P(C_k)\left[\sum_i \sum_j P(A_i = v_{ij} | C_k)^2 - \sum_i \sum_j P(A_i = v_{ij})^2\right]}{n}$$

n: no. of nodes / concepts / categories forming a partition at a given level.

$P(C_k) \sum_i \sum_j P(A_i = v_{ij} | C_k)^2$ ⇒ ~~co~~corresponds to expected no. attribute values that can be correctly guessed given a partition.

$\sum_i \sum_j P(A_i = v_{ij})^2$ ⇒ corresponds to expected no. of guesses with no such prior knowledge.

Intraclass similarity — $P(A_i = v_{ij} | C_k)$ The larger this value is the greater propotion of class members that share this attribute-value pair & the more predictable the pair is of class members.

Interclass ~~a~~ dissimilarity — $P(C_k | A_i = v_{ij})$ The larger this value is, the fewer the objects in contrasting classes that share this attribute-value pare.