ISSN: 2321-7782 (Online)
Impact Factor: 6.047

Volume 4, Issue 6, June 2016

International Journal of Advance Research in Computer Science and Management Studies

Research Article / Survey Paper / Case Study

Available online at: www.ijarcsms.com

Comparative Analysis of Various Clustering Algorithms Used in WEKA

Priyanka Sharma[1]
Dept. of Computer Science & Engg.
Guru Jambheshwar Univ. of Sci. & Technology
Hisar – India

Deepika[2]
Dept. of Computer Science and Applications
Chaudhary Devi Lal University
Sirsa – India

Abstract: ***Clustering is an unsupervised learning problem which is used to determine the intrinsic grouping in a set of unlabeled data. Grouping of objects is done on the principle of maximizing the intra-class similarity and minimizing the inter-class similarity in such a way that the objects in the same group/cluster share some similar properties/traits. There is a wide range of algorithms available for clustering. This paper presents a survey on various clustering algorithms.***

Keywords: Clustering; Data mining; WEKA.

I. INTRODUCTION

Clustering is an *unsupervised learning* technique used for grouping elements or data sets in such a way that elements in the same group are more similar (in some way or another) to each other than to those in other groups. These groups are known as clusters. Clustering[1] is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, marketing, libraries, insurance, world wide web and bioinformatics. Cluster analysis was originated in anthropology by Driver and Kroeber in 1932 and introduced to psychology by Zubin in 1938 and Robert Tryon in 1939[2][3]. Cluster analysis itself is not one specific algorithm, but the general task to be solved. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently cluster the elements. Generally used scheme used to find similarity between data elements are inter and intra- cluster distance among the cluster elements. We can show this with a simple example:

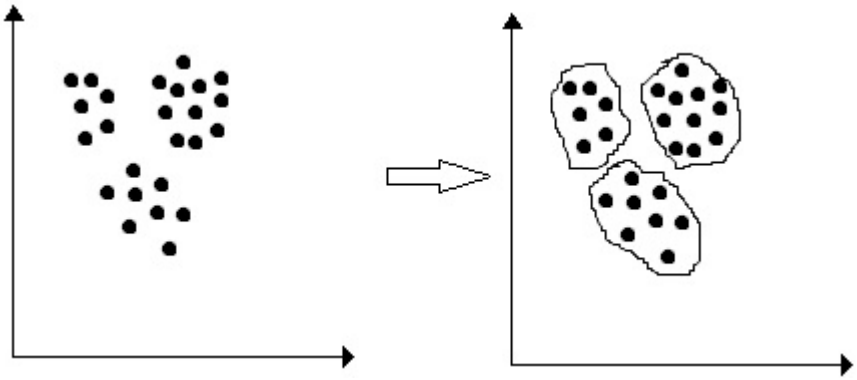

Fig 1. Clustering based on inter and intra distance measure.

In the above example, data has been divided into three clusters using the similarity criterion "*distance"*: two or more elements belong to the same cluster if they are "closer" according to a given distance. For optimizing the clusters, intra-cluster distance should be minimized and inter-cluster distance should be maximized. This clustering technique is called *distance-based clustering.*

Another kind of clustering is *conceptual clustering* in which two or more elements belong to the same cluster if they are conceptually same or similar. In other words, clusters are formed according to descriptive concepts, not according to distance measure, which is shown in the figure 2.

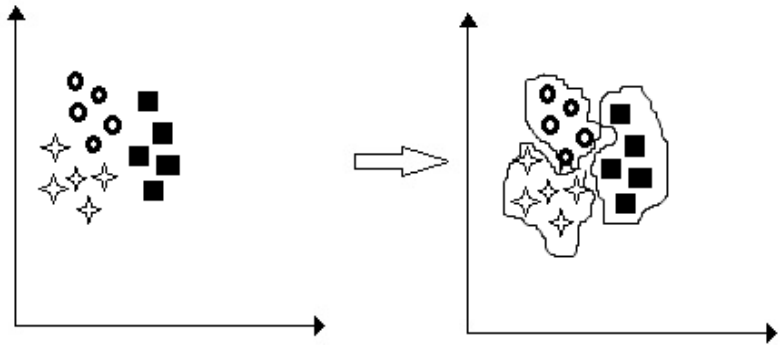

Fig 2. Clustering based on type of concepts of elements.

The appropriate clustering algorithm and parameter settings depend on the individual data set and intended use of the results. The subtle differences are often in the usage of the results: while in data mining, the resulting groups are the matter of interest, in automatic classification the resulting discriminative power is of interest.

II. Survey in Clustering Techniques

A number of clustering techniques used in data mining tool WEKA have been presented in this section. These are:

A. CLOPE- Clustering with sLOPE[4]

Like most partition-based clustering approaches, the best solution is approximated by iterative scanning of the database. However, criterion function is defined globally, only with easily computable metrics like size and width and is is very fast and scalable when clustering large transactional databases with high dimensions. To gain some basic idea behind algorithm, a small living being's database with 5 transactions {(ant, bee}, (ant, bee, crow), (ant, crow, donkey), (donkey, elephant), (donkey, elephant, fish)}. For simplicity, transaction (ant, bee) is abbreviated to ab, etc. For this small database, we want to compare the following two clustering (1) {{ab, abc, acd}, {de, def}} and (2) {{ab, abc}, {acd, de, def}}. For each cluster, the occurrence of every distinct being is counted, and then the corresponding height (H) and width (W) of the cluster is obtained. For example, cluster {ab, abc, acd} has the occurrences of a:3, b:2, c:2, and d:1, with H=2.0 ((3+2+2+1)/4 i.e. height is total number of occurrences of all the beings divided by total number of distinct ones (W)) and W=4. A larger height-to-width ratio means better intra-cluster similarity.

A transactional database D is a set of transactions {t1, ..., tn}. Each transaction is a set of items {i1, ..., im}. A clustering {C1, ... Ck} is a partition of {t1, ..., tn}, that is, $C1 \cup \ldots \cup Ck = \{t1, ..., tn\}$ and $Ci \neq \varphi$ and $Ci \cap Cj = \varphi$ for any $1 \leq i, j \leq k$. Each Ci is called a cluster and, n, m, k are used for the number of transactions, the number of items, and the number of clusters respectively.

Given a cluster C, all the distinct items (D(C)) in the cluster can be found with their respective occurrences, Occ(i, C) of item i in cluster C, that is, the number of transactions containing that item. Then the histogram of a cluster C can be drawn, with items as the X-axis, decreasingly ordered by their occurrences, and occurrence as the Y-axis for better visualization. size S(C) and width W(C) of a cluster C are defined below:

$$S(C) = \sum_{i \in D(C)} Occ(i, C) = \sum_{t_i \in C} |t_i|$$

$$W(C) = |D(C)|$$

The height of a cluster is defined as H(C)=S(C)/W(C).

It's straightforward that a larger height means a heavier overlap among the items in the cluster, and thus more similarity among the transactions in the cluster. In our running example, the height of {ab, abc, acd} is 2, the height of {acd, de, def} is 1.6, and the height of {ab, abc}and {de, def} is 1.67. So clustering (1) is better, since all the other characteristics of the two clusterings are the same. However, to define criterion function, height alone is not enough. Take a very simple database {abc, def}. There is no overlap in the two transactions, but the clustering {{abc, def}} and the clustering {{abc}, {def}} have the same height 1. Another choice works better for this example.

To define the criterion function of a clustering, the shape of every cluster as well as the number of transactions in it has been taken into account. For a clustering C = {C1, ..., Ck}, the following the criterion function has been used and found the most suitable.

$$Profit_r(\mathbf{C}) = \frac{\sum_{i=1}^{k} \frac{S(C_i)}{W(C_i)^r} \times |C_i|}{\sum_{i=1}^{k} |C_i|}$$

CLOPE is quite memory saving, even array representation of the occurrence data is practical for most transactional databases. The total memory required for item occurrences is approximately M×K×4 bytes using array of 4-byte integers, where M is the number of dimensions, and K the number of clusters. CLOPE is quite effective in finding interesting clusterings, even though it doesn't specify explicitly any inter-cluster dissimilarity metric. Moreover, CLOPE is not very sensitive to data order, and requires little domain knowledge in controlling the number of clusters.

B. Cobweb Clustering

Cobweb [5][6] is a hierarchical conceptual clustering which works without knowing in advance about the predefined number of clusters. The clusters are represented probabilistically by conditional probability P(A = v/ C) where attribute A has value v, given that the instance belongs to class C. Each node in a classification tree represents a class and is labeled by a probabilistic concept (or class) that summarizes the attribute-value distributions of objects classified under the node. The algorithm incrementally organizes observations into a classification tree. This classification tree can be used to predict missing attributes or the class of a new object. Four basic operations are employed in Cobweb which operation will be applied/ used depends on the category utility of the classification achieved by applying it. For each instance the following options (operations) are considered:

Merging Two Nodes: Merging two nodes means replacing them by a node whose children is the union of the original nodes' sets of children and which summarizes the attribute-value distributions of all objects classified under them. New instance is placed in the resulting hierarchy.

Splitting a node: A node is split by replacing it with its children into two nodes. New instance is placed in the resulting hierarchy.

Inserting a new node: A node is created corresponding to the object being inserted into the tree and instance is placed in this new node.

Passing an object down the hierarchy: Effectively calling the Cobweb algorithm on the object and the sub tree rooted in the node.

The measure used to evaluate the quality of the hierarchy/ tree is category utility (CU) measure. The motivation for the measure is highly similar to the "information gain" measure used for decision tree learning. In fact, the CU for feature-based classification is the same as the mutual information between the feature variables and the class variable, and since this measure

is much better known, mutual information is considered as the measure of category "goodness". CU attempts to maximize both the probability that two instances in the same category have attribute values in common and the probability that instances from different categories have different attribute values.

$$CU = \sum_{C}\sum_{A}\sum_{v} P(A = v)P(A = v|C)P(C|A = v)$$

P(A = v|C) is the probability that an instance has value v for its attribute A, given that it belongs to category C. The higher this probability, the more likely two instances in a category share the same attribute values.

P(C|A = v) is the probability that an instance belongs to category C, given that it has value v for its attribute A. The greater this probability, the less likely instances from different categories will have attribute values in common.

P(A = v) is a weight, assuring that frequently occurring attribute values will have stronger influence on the evaluation.

The algorithm searches the space of possible hierarchies by applying the above operators and an evaluation function based on the category utility.

C. DBSCAN - Density-based spatial clustering of applications with noise

DBSCAN [7][8] is a density-based clustering algorithm as it finds a number of clusters starting from the estimated density distribution of corresponding nodes in space. It is one of the most common clustering algorithms and also most cited in scientific literature. Outliers are handled very well in this algorithm and also it can find non-linearly separable clusters. Density is defined in terms of radius/ distance around each point and number of neighbors.

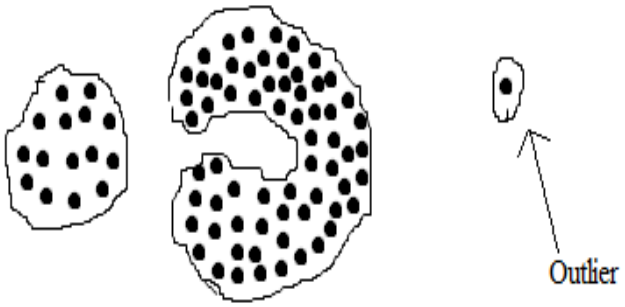

Fig 3. DBSCAN Clustering.

DBSCAN's definition of a cluster is based on the notion of density reachability. Here, these notions are represented:

Directly Density Reachable: A point p is directly density reachable from a point q if its distance from q is not greater than Ɛ (here, Ɛ is radius) and q has neighbors (points around it) greater than specified threshold.

Density Reachable: A point p is density reachable from a point q with respect to Ɛ and minimum points if there is a chain of points p1 to pn ; p1=q and pn=p such that pi+1 is directly density reachable (transitivity). It need not to be symmetric.

Density Connected: A point p is density connected to a point q with respect to Ɛ and minimum points if there is a point O such that both p and q are density reachable from O.

Cluster: A cluster C with respect to Ɛ and min points is a non-empty subset of dataset satisfying following conditions:

¥p and q: if p ₹ C and q is density reachable from p then q ₹ C.

¥p and q: if p ₹ C and q is density connected to p then q ₹ C.

Noise: If C1, C2,… ,Cn are clusters. Then, all points belonging to dataset which do not belong to any of Ci for i=1 to n are outliers.

DBSCAN's run time complexity is O(n2). The numbers of clusters need not to be specified in the data a priori, as opposed to k-means and many other algorithms. Arbitrarily/ concave shaped clusters can be found in this algorithm. However, the quality of DBSCAN depends on the distance measure used. The most common distance metric used is Euclidean distance. Especially for high-dimensional data, this metric can be rendered almost useless due to the so-called "Curse of dimensionality", making it difficult to find an appropriate value for ε. This effect is also present in any other algorithm based on Euclidean distance. Also, DBSCAN is not suitable in clustering data sets well with large differences in densities.

D. EM – Expectation Maximization Clustering [9]

Expectation–maximization (EM) [9][10] algorithm is an iterative method for finding maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models where the equations cannot be solved directly and has ability to find good estimates for any missing information (or unknown parameters) in your data at the same time. Similarly to k-means, EM selects the cluster parameters or guesses the classes of the instances, then iterate. Two phases are used in EM iteration an expectation (E) phase, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters, and maximization (M) phase, which computes parameters maximizing the expected log-likelihood found on the E step. These parameter-estimates are then used to determine the distribution of the latent variables in the next iteration E phase.

Finding a maximum likelihood solution typically requires taking the derivatives of the likelihood function with respect to all the unknown values — viz. the parameters and the latent variables — and simultaneously solving the resulting equations. Given a statistical model which generates a set A of observed data, a set of unobserved latent data or missing values x, and a vector of unknown parameters Θ, along with a likelihood function L(Θ, x, A)= P(A, x|Θ), the maximum likelihood estimate (MLE) of the unknown parameters is determined by the marginal likelihood of the observed data:

$$L(\Theta, A) = P(A|\Theta) = \sum_{x} P(A, x|\Theta)$$

However, the EM algorithm seeks to find the MLE of the marginal likelihood by iteratively applying the following two steps:

Expectation step (E step): Calculate the expected value of the log likelihood function, with respect to the conditional distribution of x given A under the current estimate of the parameters Θ(t):

$$Q\left(\Theta|\Theta^{(t)}\right) = E_{x|A,\Theta^{(t)}}[\log L(\Theta; A, x)]$$

Maximization step (M step): Find the parameter that maximizes expected log-likelihood calculated in above phase:

$$\Theta^{(t+1)} = \arg\ {}^{max}_{\Theta} Q(\Theta|\Theta^{(t)})$$

Note that in typical models to which EM is applied:

- The observed data points A may be discrete or continuous.
- The missing values x are discrete, drawn from a fixed number of values, and there is one latent variable (or missing value) per observed data point.
- The parameters are continuous, and are of two kinds: Parameters that are associated with all data points and parameters associated with a particular value of a latent variable (i.e. associated with all data points whose corresponding latent variable has a particular value).

However, it is possible to apply EM to other sorts of models. The motivation is as follows. If the value of the parameters Θ is known, the value of the latent variables x can be obtained by maximizing the log-likelihood over all possible values of x, either simply by iterating over x or through an algorithm such as the Viterbi algorithm for hidden Markov models. Conversely,

if the value of the latent variables x is known, an estimate of the parameters Θ can be found easily, typically by simply grouping the observed data points according to the value of the associated latent variable and averaging the values, or some function of the values, of the points in each group:

- First, initialize the parameters Θ to some random values.
- Compute the best value for x given these parameter values.
- Then, use the just-computed values of x to compute a better estimate for the parameters Θ.
- Iterate steps 2 and 3 until convergence.

The algorithm as just described monotonically approaches a local minimum of the cost function, and is commonly called hard EM. The k-means algorithm is an example of this class of algorithms.

However, rather than making a hard choice for x given the current parameter values and averaging only over the set of data points associated with a particular value of x, the probability of each possible value of x can be determined for each data point, and then using the probabilities associated with x can be used to find weighted average over the entire set of data points. The resulting algorithm is commonly called soft EM. The counts used to compute these weighted averages are called soft counts (as opposed to the hard counts used in a hard-EM-type algorithm such as k-means). The probabilities computed for x are posterior probabilities and are what is computed in the Expectation phase. The soft counts used to compute new parameter values are what is computed in the Maximization phase.

EM is frequently used for data clustering in machine learning, psychometrics and computer vision. With the ability to deal with missing data and observe unidentified variables, EM is becoming a useful tool to price and manage risk of a portfolio. The EM algorithm is also widely used in medical image reconstruction.

E. Farthest First Clustering [11]

Farthest first [12] is a heuristic based method of clustering. It is a variant of K Means that also chooses centroids and assigns the objects in cluster but at the point furthermost from the existing cluster centre lying within the data area. Fast clustering is provided by this algorithm in most of the cases since less reassignment and adjustment is needed.

The farthest-point [13][11] heuristic starts with an arbitrary point s1. Pick a point s2 that is as far from s1 as possible. Pick si to maximize the distance to the nearest of all centroids picked so far. That is, maximize the min {dist (si, s1), dist (si, s2), ...}. After hosing all the k representatives, partition of D can be defined as: cluster Cj consists of all points closer to sj than to any other representative. If the maximum radius of these k clusters be σ , then even in the optimal k-clustering the maximum radius must be at least σ/2 . Here optimal is in the sense of minimizing the maximum radius of any cluster.

Obviously, farthest-point heuristic based method has the time complexity O (nk), where n is number of objects in the dataset and k is number of desired clusters. In the above basic farthest-point heuristic, the first point is selected randomly. Then, the remaining k-1 points are selected deterministically. Although only one point is selected randomly, it is also hard to guarantee stable clustering results if above basic farthest-point heuristic is used in k-modes clustering. In the following, we present a method for selecting the first point deterministically.

For each Xi = [xi,1, xi,2, …, xi,m] in D that is described by m categorical attributes, f (xi,j | D) has been used to denote the frequency count of attribute value xi,j in the dataset. Then, a scoring function has been designed for evaluating each point, which is defined as:

$$Score(X_i) = \sum_{j=1}^{m} f(x_{i,j}|D)$$

In the new farthest-point heuristic, the point with highest score is selected as the first point, and remaining points are selected in the same manner as that of basic farthest-point heuristic. The rationale behind the scoring function is that, points that contain attribute values with higher frequencies are more likely to be centers. Obviously, in the new farthest-point heuristic, all points are selected deterministically. Moreover, selecting the first point according to above defined scoring function could be fulfilled in O (n) time by deploying the following procedure (with two scans over the dataset):

- In the first scan over the dataset, m hash tables are constructed as basic data structures to store the information on attribute values and their frequencies where m is number of attributes.
- In the second scan over the dataset, with the use of hashing technique, in O (1) expected time, the frequency count of an attribute value in corresponding hash table can be determined.

Therefore, the data point with largest score could be detected in O (n) time. Time complexity of the basic algorithm is O (nk), where n is number of objects in the dataset and k is number of desired clusters. Farthest-point heuristic based method is suitable for large-scale data mining applications.

F. Filtered Clusterer [14]

In mathematics, a filter [14] is a special subset of a partially ordered set. For example, the power set of some set, partially ordered by set inclusion, is a filter. Let X be a topological space and x a point of X. A filter base B on X is said to cluster at x (or have x as a cluster point) if and only if each element of B has nonempty intersection with each neighborhood of x.

- If a filter base B clusters at x and is finer than a filter base C, then C clusters at x too.
- Every limit of a filter base is also a cluster point of the base.
- A filter base B that has x as a cluster point may not converge to x. But there is a finer filter base that does. For example the filter base of finite intersections of sets of the sub base B U Nx.
- For a filter base B, the set $\cap\{cl(B0) : B0 \in B\}$ is the set of all cluster points of B (note: cl(B0) is the closure of B0). Assume that X is a complete lattice.
- The limit inferior of B is the infimum of the set of all cluster points of B.
- The limit superior of B is the supremum of the set of all cluster points of B.
- B is a convergent filter base if and only if its limit inferior and limit superior agree; in this case, the value on which they agree is the limit of the filter base.

G. Hierarchical Clustering

Hierarchical clustering [15] is a method of cluster analysis which is used to build a hierarchy of clusters. The clustering generally falls into two types:

- Agglomerative: This is a "bottom up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
- Divisive: This is a "top down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

Agglomerative Clustering: Agglomerative [16] hierarchical clustering is a bottom-up clustering method where many clusters are merged large and large clusters and so on until the optimal clusters are not obtained. It starts with taking every single object as a single cluster. Then, depending upon their similarities (distances) calculated in successive iterations, it agglomerates (merges) the closest pair of clusters (by satisfying some similarity criteria), until all of the data is in one cluster. It

should be noted that clusters generated in early stages are nested/ merged in those generated in later stages and clusters with different sizes in the tree can be useful for discovery.

- Initially, each object is assigned to a separate cluster.
- All pair-wise distances between clusters.
- Pair of clusters with the shortest distance is chosen.
- Pairs are merged (agglomerated).
- All distances from this new cluster to all other clusters is calculated.
- Repeat until no further agglomerates possible i.e. only one cluster left.

It results into a cluster tree/ dendrogram [17]. One can cut the tree at any level to produce different clustering's. It can produce an ordering of the objects, which may be informative for data display and number of clusters need not to be defined a priori. But in this clustering technique, there is no provision for relocation of objects that may have been 'incorrectly' grouped at an early stage. The result should be examined closely to ensure it makes sense. Also, use of different distance metrics for measuring distances between clusters may generate different results. Performing multiple experiments and comparing the results is recommended to support the veracity of the original results.

Divisive Clustering: Divisive clustering [18] is a top-down clustering method and is relatively less commonly used. It works in a similar way to agglomerative clustering but in the opposite direction. Initially all the objects are assumed to contain in a single cluster and then successively split into different clusters until individual objects remain in every cluster. Divisive clustering method is a descendant hierarchical algorithm which is used for clustering classical or symbolic data. This method is generally used for comparison of numerical and symbolic approach for clustering. Here is a little about the step followed in the algorithm.

1. Starts with the whole dataset.
2. At each step :
   - Choice of the cluster to be split
   - Split of this cluster
3. Stop :
   - After K iterations or
   - All clusters are singletons or have similar objects

A challenge with divisive clustering is to choose splitting/ partitioning measure as there are 2n-1 ways to partition n objects into two sub-clusters. However, to split a cluster, complete enumeration [19], heuristics [20] and Criterion optimization like split or diameter [21] etc. has been used. A tree structure called dendrogram is used to represent clusters in divisive clustering.

H. OPTICS - Ordering Points To Identify the Clustering Structure [22]

OPTICS [23] is an algorithm for finding density-based clusters in spatial data similar to DBSCAN, but the problem of detecting meaningful clusters in data of varying density is beautifully handled in this algorithm which is a shortcoming of DBSCAN. In this technique, the points of the database are arranged such that points spatially closest become neighbors. Each point contains a distance measure which stores density value and it should be greater than threshold density that needs to be accepted for a cluster in order to have both points belong to the same cluster. Dendrogram is used for the representation.

Like DBSCAN, OPTICS requires two parameters: Ɛ, the maximum distance between two points in a clusters (radius of a cluster), and MinPoints, the threshold number of points required to form a cluster. A point P is a core point if at least threshold number of neighbors/ points are found within its Ɛ- neighborhood NBƐ(P). Contrary to DBSCAN, OPTICS also considers points that are part of a more densely packed cluster, so each point is assigned a core distance that describes the distance to the MinPoints-th closest point:

$$core\_dist_{\varepsilon,MinPoints(P)} = \begin{cases} Undefined & if\ |N_\varepsilon(P)| < MinPoints \\ MinPoints - th\ smallest\ dist.to N_\varepsilon(P) & otherwise \end{cases}$$

The reachability distance of a point Q from point P is the distance between Q and P, or the core distance of P:

$$reachability\_dist_{\varepsilon,MinPoints(Q,P)} = \begin{cases} Undefined & if\ |N_\varepsilon(P)| < MinPoints \\ Max\left(core_{dist_{\varepsilon,MinPoints(P)}}, dist(P,Q)\right) & otherwise \end{cases}$$

If P and Q are nearest neighbors, then Ɛ' < Ɛ is assumed to have P and Q belong to the same cluster. To get a good clustering, value of Ɛ and Ɛ-neighborhood need to be set correctly. The parameter Ɛ is not very necessary; a maximum possible value can be set for this parameter however it plays a role with regard to complexity. It is often said that OPTICS abstracts from DBSCAN by removing this parameter, at least to the extent of only having to give the maximum value. Time complexity of the algorithm is O(n log n).

I. k-Mean Clustering

k-means [24] is one of the simplest unsupervised learning algorithms that aim to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean value. Initially, k centroids need to be chosen in the beginning. These centers should be chosen in an effective way placing them as much as possible far away from each other because different location causes different result. The next step is to take instances or points belonging to a data set and associate them to the nearest centers. After associating all the points with centers, centroids are recalculated from the clusters obtained in the process. After finding k new centroids, a new binding has to be done between the same data set points and the nearest new center. Process is repeated until no more changes are done or in other words centers do not move any more. Finally, this algorithm aims at minimizing intra cluster distance (cost function also known as squared error function), automatically inter cluster distance will be maximized.

$$Cost_{Fun} = \sum_{i=1}^{k} \sum_{p \in C_i} ||p - m_i||^2$$

where

mi – mean of ith cluster,

Ci - ith cluster and

p – point representing the object.

k-means clustering algorithm is fast, robust, relatively efficient and easier to understand. Time complexity of the algorithm is O(tknd), where n is number of objects/ points in the data set, k is number of predefined clusters, d is number of attributes/ dimension of each object, and t is the number of iterations until optimal clusters are not obtained. Best results are seen when data set are distinct or well separated from each other, if there are two highly overlapping data then k-means will not be able to resolve that there are two clusters as it can give only concave cluster. Also, algorithm requires apriori specification of the number of cluster centers. Euclidean distance measure is used as cost function which can unequally weight underlying factors. As it is a heuristic algorithm, there is no guarantee that it will converge to the global optimum and may also provide the local optima as final result depending upon initial cluster centers. Noisy data and outliers are not handled.

III. Conclusion

Clustering algorithms used in WEKA have been discussed. Any new clustering algorithm can be developed based on the study of previous ones.

References

1. M. Mor, P. Gupta, and P. Sharma, "A Genetic Algorithm Approach for Clustering."
2. K. Bailey, "Numerical taxonomy and cluster analysis," Typol. Taxon., vol. 34, p. 24, 1994.
3. R. C. Tryon, Cluster analysis: correlation profile and orthometric (factor) analysis for the isolation of unities in mind and personality. Edwards brother, Incorporated, lithoprinters and publishers, 1939.
4. Y. Yang, X. Guan, and J. You, "CLOPE: a fast and effective clustering algorithm for transactional data," in Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, 2002, pp. 682–687.
5. "Cobweb (clustering)," Wikipedia, the free encyclopedia. 06-Dec-2014.
6. I. Jonyer, D. J. Cook, and L. B. Holder, "Graph-based hierarchical conceptual clustering," J. Mach. Learn. Res., vol. 2, pp. 19–43, 2002.
7. "DBSCAN," Wikipedia, the free encyclopedia. 07-Feb-2015.
8. M. Kryszkiewicz and P. Lasek, "TI-DBSCAN: Clustering with DBSCAN by Means of the Triangle Inequality," in Rough Sets and Current Trends in Computing, 2010, pp. 60–69.
9. "Expectation–maximization algorithm," Wikipedia, the free encyclopedia. 18-Feb-2015.
10. T. K. Moon, "The expectation-maximization algorithm," Signal Process. Mag. IEEE, vol. 13, no. 6, pp. 47–60, 1996.
11. H. Zengyou, "Farthest-point heuristic based initialization methods for K-modes clustering," 2006.
12. M. Bilenko, S. Basu, and R. J. Mooney, "Integrating constraints and metric learning in semi-supervised clustering," in Proceedings of the twenty-first international conference on Machine learning, 2004, p. 11.
13. D. A. Vadeyar and H. K. Yogish, "Farthest First Clusering in Links Reorganization," Int. J. Web Semantic Technol., vol. 5, no. 3, 2014.
14. "Filter (mathematics)," Wikipedia, the free encyclopedia. 06-Dec-2014.
15. "Hierarchical clustering," Wikipedia, the free encyclopedia. 09-Feb-2015.
16. I. Davidson and S. S. Ravi, "Agglomerative hierarchical clustering with constraints: Theoretical and empirical results," in Knowledge Discovery in Databases: PKDD 2005, Springer, 2005, pp. 59–70.
17. "Dendrogram," Wikipedia, the free encyclopedia. 08-Dec-2014.
18. K. C. Gowda and T. V. Ravi, "Divisive clustering of symbolic objects using the concepts of both similarity and dissimilarity," Pattern Recognit., vol. 28, no. 8, pp. 1277–1282, 1995.
19. A. W. Edwards and L. L. Cavalli-Sforza, "A method for cluster analysis," Biometrics, pp. 362–375, 1965.
20. K. C. Gowda and G. Krishna, "Agglomerative clustering using the concept of mutual nearest neighbourhood," Pattern Recognit., vol. 10, no. 2, pp. 105–112, 1978.
21. X. Wang and H. J. Hamilton, DBRS: a density-based spatial clustering method with random sampling. Springer, 2003.
22. "OPTICS algorithm," Wikipedia, the free encyclopedia. 21-Dec-2014.
23. M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: ordering points to identify the clustering structure," in ACM Sigmod Record, 1999, vol. 28, pp. 49–60.
24. J. Han, M. Kamber, and J. Pei, Data mining: concepts and techniques. Morgan kaufmann, 2006.

AUTHOR(S) PROFILE

Priyanka Sharma, received B.Tech, M.Tech degrees in Computer Science & Engineering from Guru Jambheshwar University of Science and Technology, Hisar (Hry.) in 2012 and 2014. She is UGC-NET/JRF qualified. She worked as an assistant professor on contract basis in Chaudhary Devi Lal University, Sirsa (Hry.) from August, 2014 to Jan, 2016. She is now doing PhD from Guru Jambheshwar University of Science and Technology, Hisar (Hry.).

Deepika, received the M.C.A. degree in Computer Science and Application from Chaudhary Devi Lal University, Sirsa (Hry.) in 2013. She is UGC-NET qualified. She has been working as an assistant professor on contract basis in Chaudhary Devi Lal University, Sirsa (Hry.) since August, 2014.

