A Short Survey on Data Clustering Algorithms

Ka-Chun Wong
Department of Computer Science
City University of Hong Kong
Kowloon Tong, Hong Kong
Email: kc.w@cityu.edu.hk

Abstract—With rapidly increasing data, clustering algorithms are important tools for data analytics in modern research. They have been successfully applied to a wide range of domains; for instance, bioinformatics, speech recognition, and financial analysis. Formally speaking, given a set of data instances, a clustering algorithm is expected to divide the set of data instances into the subsets which maximize the intra-subset similarity and inter-subset dissimilarity, where a similarity measure is defined beforehand. In this work, the state-of-the-arts clustering algorithms are reviewed from design concept to methodology; Different clustering paradigms are discussed. Advanced clustering algorithms are also discussed. After that, the existing clustering evaluation metrics are reviewed. A summary with future insights is provided at the end.

I. INTRODUCTION

Nowadays, with the support of science and technology, large amounts of data has been, and will continue to be, accumulated. For example, a single human genome accounts for about four gigabytes data space [1], [2], [3] and the transaction logs in financial markets are measured in billions each day [4]. Such a large amount of data is overwhelming and prevents us from applying traditional analysis techniques. Scalable methods need to be devised to handle it. As one of the main analysis tools, cluster analysis methods have been proposed to separate the large amount of data into clusters. The data clustering methods are unsupervised which means there is not any label for model training; we do not even know the exact number of clusters beforehand. Given a set of data, a clustering method is expected to divide the data into several clusters by itself. Formally speaking, given a set of data instances, a data clustering method is expected to divide the set of data instances into the subsets which maximize the intra-subset similarity and inter-subset dissimilarity, where a similarity measure is defined beforehand.

II. CLUSTERING PARADIGMS

Since most data clustering problems have been shown to be NP-hard [5], different methods have been proposed in the past. In general, those methods can be categorized into different paradigms: Partitional Clustering, Hierarchical Clustering, Density-based Clustering, Grid-based Clustering, Correlation Clustering, Spectral Clustering, Gravitational Clustering, Herd Clustering, and Others.

A. Partitional Clustering

Data is divided into non-overlapping subsets such that each data instance is assigned to exactly one subset. For example, k-means [6] is a classical partitioning method that applies an iterative refinement approach with two main steps. The first step is to choose the means of clusters as the centroids, whereas the second step is to assign data points to their nearest centroids. In practice, its computational speed and simplicity appeal to people [7], [8]. Its main drawback is the vulnerability to its random seeding technique. In other words, if the initial seeding positions are not chosen correctly, the clustering result quality will be affected adversely.

In light of that, David Arthur and Sergei Vassilvitskii proposed a method called k-means++ [9] to improve k-means in 2007. From section 2.1 and 2.2 in [9], we can observe that the steps 2-4 of k-means++ are exactly the same as those of k-means. The main difference lies in the step 1 which is the seeding technique. A new seeding technique is proposed to replace the arbitrary seeding technique of k-mean. Given a set of seeds chosen, the seeding technique favors the data points which are far from the seeds already chosen. Thus the seeds are chosen probabilistically as dispersed as possible.

As k-means++ is the extended version of k-means method, we conducted numerical experiments to evaluate and compare their performance under 1000 replicate runs. For better visual inspection and visualization, the datasets and performance values are both depicted and tabulated in Fig. 1. We can observe that k-means++ does perform better than k-means on the first three datasets. Both the clustering score (Rand Index) and time taken have been improved. However, the performance comparison is relatively complicated on the last dataset.

![Fig. 1. Performance Comparison between k-means and k-means++.](image-url)

B. Hierarchical Clustering

Clusters are formed by following either a bottom-up approach or a top-down approach. For example, single-linkage
clustering [10] is a classic bottom-up approach in which data points are gradually agglomerated together to form clusters. In each step, all pair-wise distances are computed to identify the minimum. The parties involved in the minimal pair-wise distance are linked together. Such a step is repeated until all data points are linked together. A hierarchical tree is constructed to connect all data points at the end. A tree depth level can be chosen to cut the tree, forming clusters. To model data dynamically, a special hierarchical clustering method called Chameleon has been proposed [11]. It makes use of the inter-connectivity and closeness concept to merge and divide clusters. If the inter-connectivity and closeness between two clusters are higher than those within the clusters, then the two clusters are merged.

C. Density-based Clustering

Apart from the well-known clustering methods, there are different clustering paradigms. In density-based clustering, data is clustered based on some connectivity and density functions. For example, DBscan [12] uses density-based notions to define clusters. Two connectivity functions density-reachable and density-connected have been proposed to define each data point as either a core point or a border point. DBscan visits points arbitrarily until all points have been visited. If the point is a core point, it tries to expand and form a cluster around itself. Based on the experimental results, the authors have demonstrated its robustness toward discovering arbitrarily shaped clusters.

D. Grid-based Clustering

In grid-based clustering, the data space is divided into multiple portions (grids) at different granularity levels to be clustered individually. For example, CLIQUE [13] can automatically find subspaces with high density clusters. No data distribution assumption has been made. The empirical results demonstrated that it could scale well with the number of dimensions. Thus it is especially efficient in clustering high-dimensional data.

E. Correlation Clustering

Correlation clustering [14] was motivated from a document clustering problem in which one has a pair-wise similarity function \( f \) learned from past data. The goal is to partition the current set of documents in a way that correlates with \( f \) as much as possible. In other words, we have a complete graph of \( N \) vertices, where each edge is labeled either + or -. Our goal is to produce a partition of vertices (a clustering) that agrees with the edge labels. The authors have proved that this problem is a NP-complete problem. Hence they proposed two approximation algorithms to achieve the partitioning.

The first method called Cautious is to minimize the disagreements (number of - edges inside clusters plus the number of + edges between clusters), whereas the second method called PTAS is to maximize the agreements (number of + edges inside clusters plus the number of - edges between clusters). Basically, the ideas of the above two methods are the same (to aggregate the vertices which agree with their edge labels). The first method is discussed in detail in this work.

First, we arbitrarily choose a vertex \( v \). Then we pick up all the positive neighbors (the neighbor vertices with + edge) of the vertex and put them into a set \( A \). Having picked up all the positive neighbors of the vertex, we perform pruning. That is the 'Vertex Removal Step'. In this step, we move on to check \( 3\delta \)-bad for all the positive neighbors of the vertex, where \( \delta = 1/44 \). If there are, we remove it from the set \( A \). After the removal step, the next step is 'Vertex Addition Step' in which we try to add back some vertices which are \( 3\delta \)-good with the chosen vertex \( v \) to the set \( A \). The vertices in the set \( A \) are then chosen as one cluster. The above steps are repeated until no vertices are left or the set \( A \) becomes empty.

F. Spectral Clustering

Some of the existing clustering approaches may find local minima and require an iterative algorithm to find good clusters using different initial cluster starting points. In contrast, spectral clustering [15], [16], [17] is a relatively promising approach for clustering based on the leading eigenvectors of the matrix derived from a distance matrix. The main idea is to make use of the spectrum of the similarity matrix of the data to perform dimensionality reduction for k-means clustering in fewer dimensions. The seminal work [15] is discussed in this work.

At the beginning, we form an affinity matrix \( A \), which is a NxN matrix and \( N \) is the total number of data points. Each entry \( A_{ij} \) corresponds to the similarity measure between the data points \( s_i \) and \( s_j \). The scaling parameter \( \sigma^2 \) controls how rapidly \( A_{ij} \) falls off with the distance between \( s_i \) and \( s_j \). After we have formed the affinity matrix \( A \), we construct the Laplacian matrix \( L \) from the normalized affinity matrix of \( A \). Then we find the \( k \) leading eigenvectors (i.e. with \( k \) leading eigenvalues) of \( L \) and form the matrix \( X \) by stacking the eigenvectors in column. After we have stacked the eigenvectors to form the matrix \( X \), we normalize each row. Then we treat each row in \( X \) as a data vector and use k-means clustering algorithm to cluster them. The clustering results are projected back onto the original data (i.e. it assigns the original point \( s_i \) to cluster \( j \) if and only if row \( i \) of the matrix \( X \) is assigned to cluster \( j \)).

G. Gravitational Clustering

Distinct from the works we have mentioned, gravitational clustering is considered as a rather unique method. It was first proposed by Wright [18]. In the method, each data instance is considered as a particle within the feature space. A physical model is applied to simulate the movements of the particles. As described in [19], Jonatan et al. proposed a new gravitational clustering method using Newton laws of motion. A simplified version of gravitational clustering was proposed by Long et al. [20]. Wang et al. proposed a local shrinking method to move data toward the medians of their k nearest neighbors [21]. Blekas and Lagaris [22] proposed a similar method called Newtonian Clustering in which Newton’s equations of motion are applied to shrink and separate data, followed by Gaussian mixture model building. Molecular dynamics-like mechanism was also applied for clustering by Junlin et al [23].
H. Herd Clustering

To tackle the clustering problem, a novel clustering method, Herd Clustering (HC), has been proposed by Wong et al. [24]. It novelties lie in two aspects: (1) HC is inspired from the nature, herd behavior, which is a commonly seen phenomenon in the real world including human mobility patterns [25]. Thus it is very intuitive and easy to be understood for its good performance. (2) HC also demonstrates that cluster analysis can be done in a non-traditional way by making data alive.

HC differs from the traditional ones. Instead of trying hard to analyze data alone, it also spends effort on moving data. Two stages are proposed in HC.

Inspired by the herd behavior [26], an attraction model is used to guide data movements in the first stage. Each data instance is represented by a particle. The coordinate position of a particle is given by the values of the corresponding data instance it represents. The particles attract each other if their distances are smaller than a threshold. Each particle has its own velocity (initially zero). In each iteration, the velocity of a particle is affected by the neighborhood particles. If most particles are found in a particular direction, the velocity of the particle is accelerated toward that direction.

After all the iterations in the first stage, all data instances should be well separated and merged together. They are much easier to be clustered than before. Thus an intuitive approach is proposed to cluster data in the second stage. A list of cluster centroids is maintained. At the beginning, the centroid list is empty. For each point, we check whether its distance to any centroid is smaller than the threshold. If a centroid is detected, then the point is assigned the same cluster as the centroid. If its distances to all centroids are higher than or equal to the threshold, the point is added to the list and start a new cluster around it. After all data instances are scanned, a clustering result is obtained.

At the first glance, HC is similar to Gravitation Clustering (GC) [18]: data instances are moved according to a model. Nonetheless, their details are totally different. For instance, the model in GC is a physical model following Newton Laws of motion, while that in HC is an artificial model which is designed for computational efficiency. The particle acceleration decreases as the inter-particle distance increases in GC while they are independent in HC. Calculus is involved in GC whereas only computationally efficient operations are allowed in HC.

I. Others

There are lots of other clustering methods proposed in the past. For instance, Maulik et al. applied a genetic algorithm to search for cluster centers [27]. A globally incremental approach to k-means has been reported in [28]. Celeux et al. have proposed a novel method called Gaussian parsimonious clustering models [29]. Different distance measures have been incorporated into an objective function to cluster arbitrary number of clusters [30]. A hierarchical agglomerative clustering methodology using symbolic objects has been described in [31]. Tsao et al. used a fuzzy Kohonen network for clustering [32]. A fuzzy c-means algorithm has been developed as described in [33], [34]. An alternative pruning approach to reduce the noise effect has also been proposed for the fuzzy c-means algorithm [35]. In recent years, several kernel methods have been developed for clustering [36]. A fuzzy-rough set application to microarray data has also been reported in [37]. Hu et al. have applied a hierarchical clustering method for active learning [38]. Interestingly, Corsini et al. have trained a neural network to define dissimilarity measures which are subsequently used in the relational clustering [39]. Gullo et al. have also proposed clustering methods on uncertain data [40], [41], [42]. There are many other works; more details can be found in [10], [43], [44].

III. ADVANCED CLUSTERING

A. Clustering on Data Stream

The previous clustering methods assume data are static during clustering. Nonetheless, modern data are not static necessarily. In fact, data can be transmitted in streaming form; for instance, real-time financial stock market data, video surveillance data, and social media data. Modern data keeps itself changing and evolving during the course of clustering. For analysis of such data, the ability to process the data in a timely manner with little memory is crucial. In light of that, different data stream clustering methods are proposed. For instance, Guha et al. have proposed one of the first-known method, STREAM, to solve the k-median problem on streaming data with constant-factor approximation [45]. An incremental clustering method (COBWEB) has also been proposed to maintain a hierarchical clustering tree on streaming data by Fisher [46]. Zhang et al. have proposed an efficient data clustering method for large datasets [47]. Thanks to its linear complexity and single-pass nature, it can also be applied to cluster data streams with a tree data structure, CF Tree [47]. On the other hand, an incremental clustering method (C2ICM) has been proposed to data stream clustering problems. In particular, a lower bound for its clustering performance has also been provided [48].

B. Clustering on Sequence Data

In the past years, probabilistic graphical models have been successfully applied to different problems such as gene clustering [49], [50], [51]. In particular, Hidden Markov Model (HMM) has been demonstrated successful for clustering sequence data in a wide range of domains [52].

1) Description: Hidden Markov Model (HMM) is a probabilistic graphical model which assumes a sequence of symbols is controlled and generated by a corresponding sequence of hidden states with the same sequence length. In particular, Markov property is assumed for the sequence of hidden states; in other words, each hidden state solely depends on its previous hidden state on the same sequence. Although such Markov assumption over-simplifies the independence between different states, it can work fairly well in practice. Moreover, it greatly reduces the computational complexity in HMM learning and
inference. Mathematically, an HMM can be described as $\theta$:

\[
\theta = (\{a_{ij}\}, \{b_i(x)\}, \{\pi_i\})
\]

$\forall i, j \in \{1, 2, ..., N\}; \forall x \in X$

subject to

\[
\sum_{i=1}^{N} \pi_i = 1
\]

\[
\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i \in \{1, 2, ..., N\}
\]

\[
\sum_{x \in X} b_i(x) = 1 \quad \forall i \in \{1, 2, ..., N\}
\]

\[
0 \leq \pi_i, a_{ij}, b_i(x) \leq 1 \quad \forall i, j \in \{1, 2, ..., N\}, \forall x \in X
\]

where $a_{ij}$ is the transition probability from state $i$ to state $j$; $b_i(x)$ is the emission probability to emit $x$ at state $i$; $\pi_i$ is the initial state probability for state $i$.

For illustrative purposes, an HMM example with $N = 3$ hidden states is depicted in Figure 2. In that HMM example, we have 3 hidden states. At the beginning of the sequence, we have the initialization probabilities $\{\pi_1, \pi_2, \pi_3\}$ for each hidden state, representing their chances to be the first hidden state. After that, the transition probabilities $\{a_{ij}\}$ determine the next hidden state recursively. For each hidden state traversal, depending on the current state, a symbol $x$ is emitted and appended to form an output sequence based on the emission probabilities $\{b_i(x)\}$.

2) Model Learning: To learn an HMM from sequences, Baum-Welch algorithm is usually applied to learn the unknown parameters [52]. Mathematically, Baum-Welch algorithm is an Expectation Maximization (EM) algorithm to find the maximal likelihood estimates of the HMM parameters. Thus we would like to note that Baum-Welch algorithm highly depends on the first random initialization iteration and can be trapped in local optima. Multiple runs are usually adopted to circumvent such issues. Mathematically, the Baum-Welch training algorithm can be described herein:

**Input:** A set of sequences $S = \{s_1, s_2, s_3, ..., s_M\}$ of length $L$. Each sequence $s_m$ can be represented as $s_m = s_{m1}s_{m2}...s_{mL}$ where $s_{mp} \in X \forall m \in \{1, 2, ..., M\}, \forall p \in \{1, 2, ..., L\}$.

**Output:** An HMM model $\theta$ trained to represent the set of sequences:

\[
\theta = (\{a_{ij}\}, \{b_i(x)\}, \{\pi_i\})
\]

$\forall i, j \in \{1, 2, ..., N\}; \forall x \in X$

where $a_{ij}$ is the transition probability from state $i$ to state $j$; $b_i(x)$ is the emission probability to emit $x$ at state $i$; $\pi_i$ is the initial state probability for state $i$.

**Method:** At the beginning of Baum-Welch algorithm, we randomly initialize those HMM model parameters $\theta_0$ and iteratively refine them in each iteration. In the expectation step (E-step) of the $l$-th iteration, we calculate the expected values of being in state $i$ based on the current parameter estimates $\theta_l$. Specifically, we calculate:

\[
\gamma^m_p(i) = \frac{\alpha^m_p(i)\beta^m_p(i)}{P(s_m; \theta_l)}
\]

$\forall m \in \{1, 2, ..., M\}, \forall p \in \{1, 2, ..., L\}, \forall i \in \{1, 2, ..., N\}$

where $\gamma^m_p(i)$ is the expected probability of being in state $i$ at the $p$-th position of the $m$-th sequence $s_m$; $\alpha^m_p(i)$ and $\beta^m_p(i)$ are the forward and backward probability of the $m$-th sequence $s_m$ to be in state $i$ at the $p$-th position as calculated by the dynamic programming approach [52]. $P(s_m; \theta_l)$ is the probability of observing $s_m$ given the existing HMM model parameter $\theta_l$ which can be calculated as $P(s_m; \theta_l) = \sum_{i=1}^{N} \alpha^m_p(i)\beta^m_p(i)$. In addition, we also calculate the expected values of state transitions from state $i$ to state $j$:

\[
\zeta^m_p(i, j) = \frac{\alpha^m_p(i)a_{ij}b_j(s_{mp}\beta^m_p+1(j))}{P(s_m; \theta_l)}
\]

$\forall m \in \{1, 2, ..., M\}, \forall p \in \{1, 2, ..., L\}, \forall i, j \in \{1, 2, ..., N\}$

where $\zeta^m_p(i, j)$ is the expected probability of transitioning from state $i$ at the $p$-th position to state $j$ at the $(p+1)$-th position for the $m$-th sequence $s_m$, given the current parameter estimates $\theta_l$.

In the maximization step (M-step) of the $l$-th iteration, those model parameters are refined to be the maximal likelihood estimates for those expected values:

\[
\pi'_i = \frac{\sum_{m=1}^{M} \gamma^m_p(i)}{M} \quad \forall i \in \{1, 2, ..., N\}
\]

\[
a'_{ij} = \frac{\sum_{m=1}^{M} \sum_{p=1}^{L-1} \zeta^m_p(i, j)}{\sum_{m=1}^{M} \sum_{p=1}^{L} \gamma^m_p(i)} \quad \forall i, j \in \{1, 2, ..., N\}
\]

\[
b'_i(x') = \frac{\sum_{m=1}^{M} \sum_{p=1}^{L} \gamma^m_p(i)|s_{mp} = x|}{\sum_{m=1}^{M} \sum_{p=1}^{L} \gamma^m_p(i)} \quad \forall i \in \{1, 2, ..., N\}, \forall x' \in \{A, C, G, T\}
\]

\[
\theta_{l+1} = (\{a'_{ij}\}, \{b'_i(x')\}, \{\pi'_i\})
\]

$\forall i, j \in \{1, 2, ..., N\}; \forall x' \in \{A, C, G, T\}$

The new HMM model parameters $\theta_{l+1}$ are used in the next iteration. We repeat the E-step and M-step alternatively until the HMM model parameters are not changed anymore. In other words, the difference between $\theta_l$ and $\theta_{l+1}$ converges to a numerically negligible value at which a local optimum is found.

IV. VERIFICATION

A. Benchmark Data Sources

Benchmark datasets can be downloaded from the UCI Machine Learning Repository [53].

B. Performance Metrics for Clustering

For clustering, Rand Index [54], Purity [55], F-measure [56], and Normalized Mutual Information (NMI) [56] are usually adopted for performance benchmarking. Rand Index is based on the intra-cluster similarity and inter-cluster dissimilarity. For the intra-cluster similarity, if a pair of data vectors is assigned the same cluster in both the target result and the clustering result, then the score will be increased by one. For
the inter-cluster dissimilarity, if a pair of vectors is assigned
different clusters in both the target result and the clustering
result, then the score will also be increased by one. On the
contrary, if a pair of data vectors is in the same cluster in
the target result, but not in the clustering result, the score
will not be increased. After we have checked all the possible
pairs, the score is normalized by the total number of possible pairs.
Mathematically, the formula is derived as follows

\[
\text{Rand Index} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} s_{ij}}{n^2 - n} \text{ where } i \neq j,
\]

\[
s_{ij} = \begin{cases} 
1 & \text{if } G_o(d_i) = G_o(d_j) \text{ and } G(d_i) = G(d_j), \\
1 & \text{if } G_o(d_i) \neq G_o(d_j) \text{ and } G(d_i) \neq G(d_j), \\
0 & \text{otherwise.}
\end{cases}
\]

where \( n \) is the number of data vectors, \( d_i \) is the \( i \)th data
erator, \( d_j \) is the \( j \)th data vector, \( G_o(d) \) is the cluster
group id of a data vector \( d \) in the target result, \( G(d) \) is the
cluster group id of a data vector \( d \) in the clustering result. On the other
hand, F-measure is similar to Rand Index with the exception
that true negatives are not taken into account. Mathematically,
the formula is derived as follows:

\[
F - \text{measure} = \frac{2a}{2a + b + c}
\]

\[
a = \sum_{i=1}^{n} \sum_{j=1}^{n} [i \neq j][G_o(d_i) = G_o(d_j) \& G(d_i) = G(d_j)].
\]

\[
b = \sum_{i=1}^{n} \sum_{j=1}^{n} [i \neq j][G_o(d_i) \neq G_o(d_j) \& G(d_i) = G(d_j)].
\]

\[
c = \sum_{i=1}^{n} \sum_{j=1}^{n} [i \neq j][G_o(d_i) = G_o(d_j) \& G(d_i) \neq G(d_j)].
\]

, where \([...]\) is the Iverson bracket. In contrast, purity solely
measures the intra-cluster similarity. Nevertheless, it is useful
in the sense that we only care about the quality of individual
clusters. Mathematically, the purity of a cluster \( C_i \) of size \( n_i \)
is defined below. For \( n \) data instances with \( k \) cluster groups,
the overall purity of a clustering result is defined as:

\[
P(C_i) = \frac{1}{n_i} \max_{j} (n_i^j)
\]

\[
Purity = \frac{1}{n} \sum_{i=1}^{k} \frac{n_i}{n} P(C_i)
\]

, where \( n_i^j \) is the number of the data instances of \( j \)th class
that are assigned to the \( i \)th cluster. To account for all the
performance results, Normalized Mutual Information (NMI)
can also be used [56]. For all non-deterministic methods,
the performance metrics are taken by averaging over multiple runs.
For all deterministic methods, the performance metrics are
taken by running once only.

C. Performance Metrics for Prediction

From the perspective of predictive tasks, a clustering out-
come can be categorized into 4 types. If the clustering outcome
is consistent with the truth, it is called either True Positive
\((TP)\) or True Negative \((TN)\), depending on the actual value.
Otherwise, it is called False Positive \((FP)\) or False Negative
\((FN)\) respectively. In different problem domains, FPs and TNs
are depreciated and weighted differently. For instance, FPs are
more tolerated than FNs in human disease diagnosis.

To summarize the prediction performance of a clustering
method, accuracy is widely adopted. It is defined as follows:

\[
\text{Accuracy} = \frac{TPs + NPs}{TPs + FNs + FPs + TNs}
\]

Nonetheless, accuracy may be non-informative if the dataset is
imbalanced or mis-clustering cost is very high. For instance, if
only the performance of a method on positive class prediction
is practically interesting, we can adopt precision and sensitivity
(a.k.a. true positive rate and recall) which are defined as follows:

\[
\text{Precision} = \frac{TPs}{TPs + FPs}
\]

\[
\text{Sensitivity} = \frac{TPs}{TPs + FNs}
\]

Alternatively, F-measure can be applied to combine precision
and sensitivity into a single performance metric. It is defined as
the harmonic mean of precision and sensitivity. The duals
of precision and sensitivity for negative class clustering are
negative predictive value \((NPV)\) and specificity respectively.

\[
NPV = \frac{TNs}{TNs + FPs}
\]

\[
\text{Specificity} = \frac{TNs}{TNs + FNs}
\]

In particular, we would like to note that the well-known false
positive rate \((FPR)\) and false discovery rate \((FDR)\) are defined as
follows:

\[
FPR = 1 - \text{Specificity} \quad FDR = 1 - \text{Precision}
\]

Although the performance metrics described are very suitable
for evaluating discrete clustering predictions. Nonetheless, the
modern clustering methods usually assign a confidence value
to each of its prediction. To examine the modern methods in
full spectrum, receiver operating characteristics (ROC) curves
and precision-recall (PRC) curves are proposed. Different
thresholds are cut at the confidence values to observe the
performance trade-off of each method. For instance, the trade-
off between sensitivity and false positive rates can be observed
from ROC curves whereas that between precision and recall
can be observed from PRC curves. The area under ROC curves
\((AUC)\) is usually adopted as a benchmarking metric.

D. Evaluation Procedures

The most typical evaluation procedure is to divide a dataset
into two sets: training dataset and testing dataset. The training
dataset is used for training a clustering model, while the testing
dataset is isolated and reserved for testing the trained model.
In particular, the most common procedure is N-fold cross-
validation which has \( N \) iterations. The dataset is randomly
divided into \( N \) non-overlapping subsets. In each iteration, a
subset is rotated as the testing dataset while the others are
assigned as the corresponding training dataset. If the input
data is scarce or costly, leave-one-out cross-validation can also
be applied. In that case, only one data sample is left out for
testing, while the others are allocated as the training dataset
in each iteration.
E. Statistical Tests

Since some of the existing clustering methods are stochastic, multiple replicate runs need to be executed for comprehensive benchmarking [24]. The means and standard deviations of performance metrics are usually reported for fair comparison. To justify the results, statistical tests are adopted to assess the statistical significances; For instance, t-tests, Mann-Whitney U-tests (MWU), and Kolmogorov-Smirnov test (KS).

V. Benchmarking

To investigate the performance difference between those methods, four representative methods have been selected and run on different datasets. K-means++ is chosen for its simplicity and superior performance over the traditional k-means method; Correlation clustering is selected to represent the algorithms with solid theoretical support; Unsupervised optimal fuzzy clustering is chosen to represent the soft clustering algorithms; Spectral clustering is selected to represent the modern clustering algorithms. Since all of the methods selected are stochastic, 100 replicate runs are executed to compute the average performance metrics for each method on each dataset. All the parameters were tuned for each algorithm and dataset manually. The results are depicted in Fig. 3.

From the results, we can observe that the clustering methods exhibit different characteristics on different datasets. In general, based on the performance metric (Rand Index), spectral clustering is found to perform the best among the selected algorithms whereas the performance of correlation clustering is relatively limited. Based on the time taken, k-means++ is the fastest one, whereas correlation clustering is the slowest one on most datasets. The top three datasets are the most typical datasets. Each cluster forms a globular shape. It is not hard for us to expect that they can be solved by most clustering algorithms. The result turns out to concede with our expectation, except correlation clustering. The middle three datasets are difficult datasets. Each cluster is an irregular shape. Within the same dataset, each cluster is even not guaranteed to be similar to the other clusters. Interestingly, a nearly perfect result can be obtained by spectral clustering, reflecting that the dimensional transformation ability within spectral clustering does play a role in lowering the difficulties in handling such irregular data shapes. The bottom four datasets are the well-known datasets taken from the UCI machine learning repository. The number of attributes is ranged from 4 to 32. The number of class labels is ranged from 2 to 10. The number of instances is ranged from 150 to 1484. In the experiment, each algorithm has managed to perform well on a particular dataset. No conclusive insights can be drawn from the result. The data dependency of the clustering algorithms is fully reflected on those datasets.

VI. Summary and Future Works

A. Summary

With growing data, cluster algorithms (also known as cluster analysis) become important tools for analyzing data. In this book chapter, we have reviewed the existing clustering algorithms from different paradigms: Partitional Clustering, Hierarchical Clustering, Density-based Clustering, Grid-based Clustering, Correlation Clustering, Spectral Clustering, Gravitational Clustering, Herd Clustering, and Others. Especially, we have focused on their methodologies and design concepts. Advanced clustering methods have also been reviewed; for instance, data stream clustering and sequence clustering.

To verify the algorithms’ competitiveness, different types of performance metrics have been defined and reviewed. In particular, benchmark studies have been conducted to observe the empirical performance of the selected methods: k-means++, correlation clustering, fuzzy clustering, and spectral clustering. The numerical results reveal that spectral clustering has its own competitive edge over the other methods on low-dimensional datasets. For high-dimensional datasets, we cannot observe any significant performance difference between the selected methods.

Nonetheless, during the course of the studies here, we found several future directions which we believe they are promising. They are described in the following section.

B. Future Works

1) Computational Scalability: As mentioned at the very beginning of this book chapter, the recent advancements of science and technologies enable massive data generation in recent years. Some of the existing computational methods may not scale with the large amount of data. For instance, the high computational complexity of spectral clustering method [57] is no longer practical to be run on the current datasets. It is imperative for us to develop new and scalable methods to keep in pace with the data generation speed.

2) Advanced Learning Methods: In this book chapter, we have provided an overview on clustering. It is undeniable that other machine learning methods can be applied as well [58]; for instance, probabilistic graphical models can be developed and applied to capture/eliminate the uncertainty and noises in real world data.

3) Domain Knowledge: The existing clustering algorithms are built for general purposes. Domain knowledge can be incorporated if a clustering algorithm is applied to a specific task; for instance, if data is sparse, a sparse clustering algorithm can be applied to boost up the execution speed.

REFERENCES


Fig. 2. Hidden Markov Model (HMM) example with $N = 3$ hidden states.
<table>
<thead>
<tr>
<th>Dataset</th>
<th>100 runs</th>
<th>k-means++</th>
<th>Correlation Clustering</th>
<th>Unsupervised Optimal Fuzzy Clustering</th>
<th>Spectral Clustering</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Setting</td>
<td>K=2, T=40</td>
<td>K=2, stopping threshold = 0.1, g = 0.2</td>
<td>K=2, o = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rand Index</td>
<td>0.93362</td>
<td>0.50482</td>
<td>0.93593</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time (sec)</td>
<td>7.1</td>
<td>28427</td>
<td>22.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2240.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Setting</td>
<td>K=6, T=30</td>
<td>K=6, stopping threshold = 0.1, g = 0.2</td>
<td>K=6, o = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rand Index</td>
<td>0.99291</td>
<td>0.79183</td>
<td>0.99312</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time (sec)</td>
<td>10.1</td>
<td>1053.9</td>
<td>101.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3788.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Setting</td>
<td>K=5, T=20</td>
<td>K=5, stopping threshold = 0.1, g = 15</td>
<td>K=5, o = 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rand Index</td>
<td>0.93335</td>
<td>0.77976</td>
<td>0.92286</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time (sec)</td>
<td>9.9</td>
<td>1174.3</td>
<td>51.2500</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>606</td>
</tr>
<tr>
<td>Iris dataset</td>
<td></td>
<td>Setting</td>
<td>K=2, T=50</td>
<td>K=2, stopping threshold = 0.1, g = 0.2</td>
<td>K=2, o = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rand Index</td>
<td>0.50147</td>
<td>0.52527</td>
<td>0.50197</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time (sec)</td>
<td>14.8</td>
<td>18642</td>
<td>57.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2955.2</td>
</tr>
<tr>
<td>Wine dataset</td>
<td></td>
<td>Setting</td>
<td>K=2, T=43</td>
<td>K=2, stopping threshold = 0.1, g = 0.2</td>
<td>K=2, o = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rand Index</td>
<td>0.49968</td>
<td>0.53187</td>
<td>0.50091</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time (sec)</td>
<td>6.2</td>
<td>10141</td>
<td>20.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>929.1</td>
</tr>
<tr>
<td>Yeast dataset</td>
<td></td>
<td>Setting</td>
<td>K=2, T=80</td>
<td>K=2, stopping threshold = 0.1, g = 0.2</td>
<td>K=2, o = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rand Index</td>
<td>0.51742</td>
<td>0.50304</td>
<td>0.51805</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time (sec)</td>
<td>19.4</td>
<td>66962</td>
<td>151.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9489.0</td>
</tr>
<tr>
<td>Breast Cancer Wisconsin (Diagnosis) dataset</td>
<td></td>
<td>Setting</td>
<td>K=3, T=25</td>
<td>K=3, stopping threshold = 0.1, g = 0.1</td>
<td>K=3, o = 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rand Index</td>
<td>0.70915</td>
<td>0.66979</td>
<td>0.71035</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time (sec)</td>
<td>1.1719</td>
<td>126.5781</td>
<td>17.2188</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>22.6719</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Setting</td>
<td>K=10, T=0.2</td>
<td>K=10, stopping threshold = 0.1, g = 0.2</td>
<td>K=10, o = 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rand Index</td>
<td>0.74245</td>
<td>0.7616</td>
<td>0.72487</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time (sec)</td>
<td>99.1875</td>
<td>8248.6</td>
<td>32.3281</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3631.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Setting</td>
<td>K=2, T=1000</td>
<td>K=2, stopping threshold = 0.1, g = 10</td>
<td>K=2, o = 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rand Index</td>
<td>0.75038</td>
<td>0.50586</td>
<td>0.75038</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time (sec)</td>
<td>2.6875</td>
<td>1399.0</td>
<td>63.6094</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>238.3906</td>
</tr>
</tbody>
</table>

Fig. 3. Performance Comparison between the methods selected.