Clustering and the Continuous *k*-Means Algorithm

Vance Faber

Many types of data analysis, such as the interpretation of Landsat images discussed in the accompanying article, involve datasets so large that their direct manipulation is impractical. Some method of data compression or consolidation must first be applied to reduce the size of the dataset without losing the essential character of the data. All consolidation methods sacrifice some detail; the most desirable methods are computationally efficient and yield results that are—at least for practical applications—representative of the original data. Here we introduce several widely used algorithms that consolidate data by clustering, or grouping, and then present a new method, the continuous *k*-means algorithm,* developed at the Laboratory specifically for clustering large datasets.

Clustering involves dividing a set of data points into non-overlapping groups, or clusters, of points, where points in a cluster are "more similar" to one another than to points in other clusters. The term "more similar," when applied to clustered points, usually means closer by some measure of proximity. When a dataset is clustered, every point is assigned to some cluster, and every cluster can be characterized by a single reference point, usually an average of the points in the cluster. Any particular division of all points in a dataset into clusters is called a partitioning.

One of the most familiar applications of clustering is the classification of plants or animals into distinct groups or species. However, the main purpose of clustering Landsat data is to reduce the size and complexity of the dataset. Data reduction is accomplished by replacing the coordinates of each point in a cluster with the coordinates of that cluster's reference point. Clustered data require considerably less storage space and can be manipulated more quickly than the original data. The value of a particular clustering method will depend on how closely the reference points represent the data as well as how fast the program runs.

A common example of clustering is the consolidation of a set of students' test scores, expressed as percentages, into five clusters, one for each letter grade A, B, C, D, and F (see Figure 1). The test scores are the data points, and each cluster's reference point is the average of the test scores in that cluster. The letter grades can be thought of as symbolic replacements for the numerical reference points.

Test scores are an example of one-dimensional data; each data point represents a single measured quantity. Multidimensional data can include any number of measurable attributes; a biologist might use four attributes of duck bills (four-dimensional data: size, straightness, thickness, and color) to sort a large set of ducks into several species. Each independent characteristic, or measurement, is one dimension. The consolidation of large, multidimensional datasets is the main pur-

* The continuous *k*-means algorithm is part of a patented application for improving both the processing speed and the appearance of color video displays. The application is commercially available for Macintosh computers under the names *Fast Eddie*, ©1992 and *Planet Color*, ©1993, by Paradigm Concepts, Inc., Santa Fe, NM. This software was developed by Vance Faber, Mark O. Mundt, Jeffrey S. Saltzman, and James M. White.

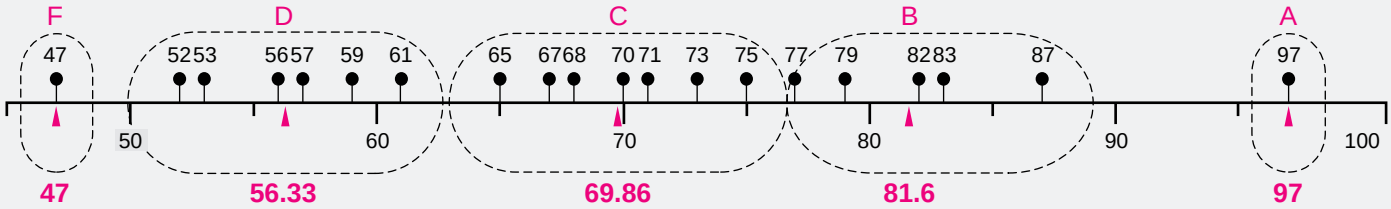

Figure 1. Clustering Test Scores
The figure illustrates an arbitrary partitioning of 20 test scores into 5 non-overlapping clusters (dashed lines), corresponding to 5 letter grades. The reference points (means) are indicated in red.

pose of the field of cluster analysis. We will describe several clustering methods below. In all of these methods the desired number of clusters k is specified beforehand. The reference point z_i for the cluster i is usually the centroid of the cluster. In the case of one-dimensional data, such as the test scores, the centroid is the arithmetic average of the values of the points in a cluster. For multidimensional data, where each data point has several components, the centroid will have the same number of components and each component will be the arithmetic average of the corresponding components of all the data points in the cluster.

Perhaps the simplest and oldest automated clustering method is to combine data points into clusters in a pairwise fashion until the points have been condensed into the desired number of clusters; this type of agglomerative algorithm is found in many off-the-shelf statistics packages. Figure 2 illustrates the method applied to the set of test scores given in Figure 1.

There are two major drawbacks to this algorithm. First—and absolutely prohibitive for the analysis of large datasets—the method is computationally inefficient. Each step of the procedure requires calculation of the distance between every possible pair of data points and comparison of all the distances. The second difficulty is connected to a more fundamental problem in cluster analysis: Although the algorithm will always produce the desired number of clusters, the centroids of these clusters may not be particularly representative of the data.

What determines a "good," or representative, clustering? Consider a single cluster of points along with its centroid or mean. If the data points are tightly clustered around the centroid, the centroid will be representative of all the points in that cluster. The standard measure of the spread of a group of points about its mean is the variance, or the sum of the squares of the distance between each point and the mean. If the data points are close to the mean, the variance will be small. A generalization of the variance, in which the centroid is replaced by a reference point that may or may not be a centroid, is used in cluster analysis to indicate the overall quality of a partitioning; specifically, the error measure E is the sum of all the variances:

$$E = \sum_{i=1}^{k} \sum_{j=1}^{n_i} \| x_{ij} - z_i \|^2,$$

where x_{ij} is the jth point in the ith cluster, z_i is the reference point of the ith cluster, and n_i is the number of points in that cluster. The notation $\|x_{ij} - z_i\|$ stands for the distance between x_{ij} and z_i. Hence, the error measure E indicates the overall spread of data points about their reference points. To achieve a representative clustering, E should be as small as possible.

The error measure provides an objective method for comparing partitionings as well as a test for eliminating unsuitable partitionings. At present, finding the best

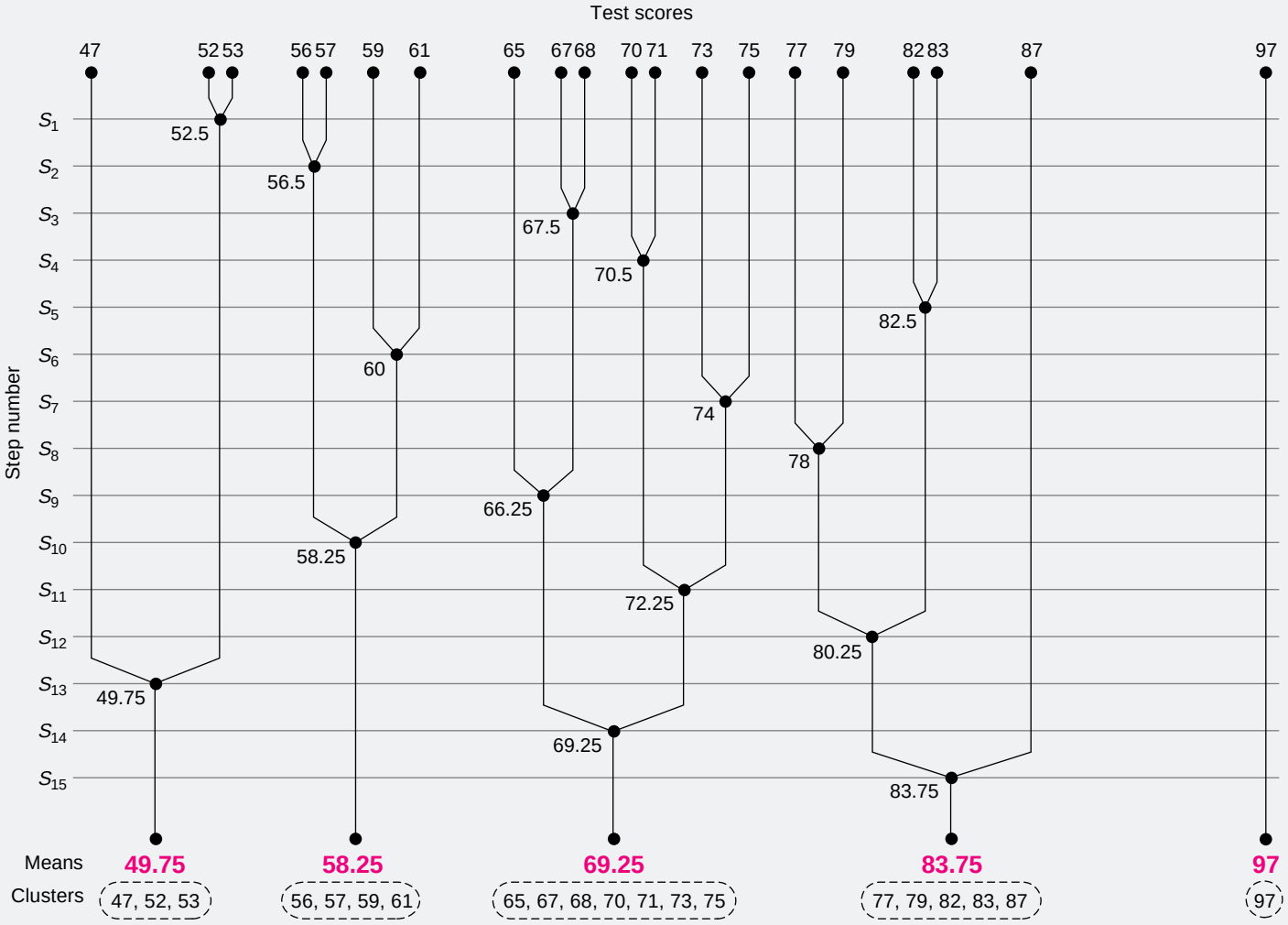

Figure 2. Pairwise Agglomerative Clustering

The figure illustrates the operation of an agglomerative clustering method, in which the 20 test scores of Figure 1 are successively merged by pairs of points and/or pairs of clusters until all the scores are collected into 5 clusters. The steps of the algorithm are shown in the branching of a dendrogram, or tree structure (much like a genealogy). A node, or branch point, indicates the merging of two branches into one, i.e. two data points into one cluster, or two clusters into one larger cluster. The algorithm begins with 20 separate clusters of one point apiece. For the first step in the algorithm, the closest two points (here, scores of 52 and 53) are found and merged into one cluster {52,53}. The two individual points are replaced by a single point equal to the unweighted average of the two points (52.5). The next step repeats this process (find the closest two points, calculate the average, merge the points), but with 19 points and 19 clusters (18 one-point clusters, plus 1 two-point cluster). There will be only one new branch, or merge at each step. Hence, if there is more than one pair of points at the minimum distance, only one pair will be merged at each step. It takes 15 steps to consolidate 20 points into 5 clusters.

partitioning (the clustering most representative of an arbitrary dataset) requires generating all possible combinations of clusters and comparing their error measures. This can be done for small datasets with a few dozen points, but not for large sets—the number of different ways to combine 1 million data points into 256 clusters, for example, is $256^{1,000,000}/256!$, where 256! is equal to $256 \times 255 \times 254 \times \cdots \times 2 \times 1$. This number is greater than $10^{2,000,000}$, or 1 followed by 2 million zeros.

When clustering is done for the purpose of data reduction, as in the case of the Landsat images, the goal is not to find the best partitioning. We merely want a reasonable consolidation of N data points into k clusters, and, if necessary, some efficient way to improve the quality of the initial partitioning. For that purpose, there is a family of iterative-partitioning algorithms that is far superior to the agglomerative algorithm described above.

Iterative algorithms begin with a set of k reference points whose initial values are usually chosen by the user. First, the data points are partitioned into k clusters: A data point x becomes a member of cluster i if z_i is the reference point closest to x. The positions of the reference points and the assignment of the data points to clusters are then adjusted during successive iterations. Iterative algorithms are thus similar to fitting routines, which begin with an initial "guess" for each fitted parameter and then optimize their values. Algorithms within this family differ in the details of generating and adjusting the partitions. Three members of this family are discussed here: Lloyd's algorithm, the standard *k*-means algorithm, and a continuous *k*-means algorithm first described in 1967 by J. MacQueen and recently developed for general use at Los Alamos.

Conceptually, Lloyd's algorithm is the simplest. The initial partitioning is set up as described above: All the data points are partitioned into k clusters by assigning each point to the cluster of the closest reference point. Adjustments are made by calculating the centroid for each of those clusters and then using those centroids as reference points for the next partitioning of all the data points. It can be proved that a local minimum of the error measure E corresponds to a "centroidal Voronoi" configuration, where each data point is closer to the reference point of its cluster than to any other reference point, and each reference point is the centroid of its cluster. The purpose of the iteration is to move the partition closer to this configuration and thus to approach a local minimum for E.

For Lloyd's and other iterative algorithms, improvement of the partitioning and convergence of the error measure E to a local minimum is often quite fast—even when the initial reference points are badly chosen. However, unlike guesses for parameters in simple fitting routines, slightly different initial partitionings generally do not produce the same set of final clusters. A final partitioning will be better than the initial choice, but it will not necessarily be the best possible partitioning. For many applications, this is not a significant problem. For example, the differences between Landsat images made from the original data and those made from the clustered data are seldom visible even to trained analysts, so small differences in the clustered data are even less important. In such cases, the judgment of the analyst is the best guide as to whether a clustering method yields reasonable results.

(a) Setup:
Reference point 1 (filled red circle) and reference point 2 (filled black circle) are chosen arbitrarily. All data points (open circles) are then partitioned into two clusters: each data point is assigned to cluster 1 or cluster 2, depending on whether the data point is closer to reference point 1 or 2, respectively.

(b) Results of first iteration:
Next each reference point is moved to the centroid of its cluster. Then each data point is considered in the sequence shown. If the reference point closest to the data point belongs to the other cluster, the data point is reassigned to that other cluster, and both cluster centroids are recomputed.

(c) Results of second iteration:
During the second iteration, the process in Figure 3(b) is performed again for every data point. The partition shown above is stable; it will not change for any further iteration.

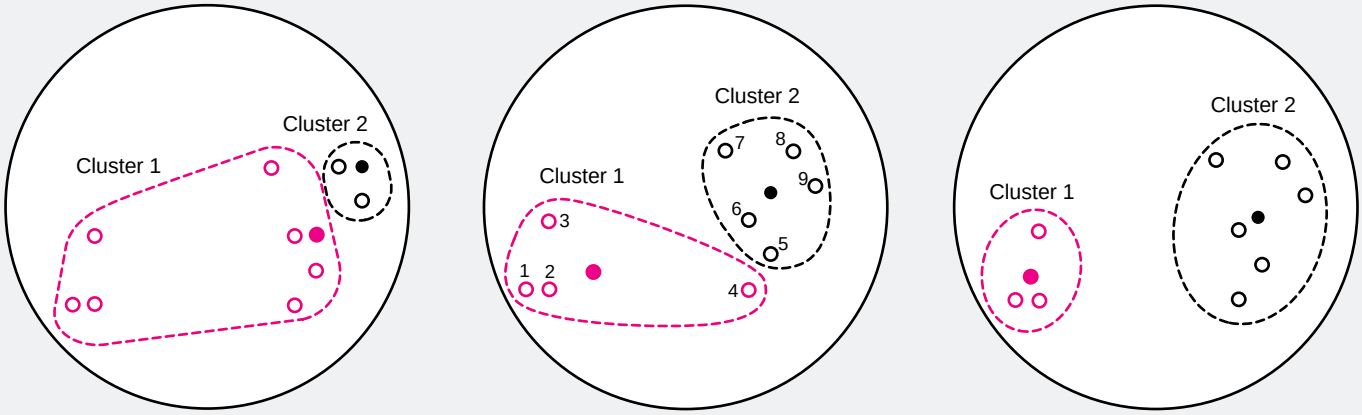

Figure 3. Clustering by the Standard *k*-Means Algorithm
The diagrams show results during two iterations in the partitioning of nine two-dimensional data points into two well-separated clusters, using the standard *k*-means algorithm. Points in cluster 1 are shown in red, points in cluster 2 are shown in black; data points are denoted by open circles and reference points by filled circles. Clusters are indicated by dashed lines. Note that the iteration converges quickly to the correct clustering, even for this bad initial choice of the two reference points.

The standard *k*-means algorithm differs from Lloyd's in its more efficient use of information at every step. The setup for both algorithms is the same: Reference points are chosen and all the data points are assigned to clusters. As with Lloyd's, the *k*-means algorithm then uses the cluster centroids as reference points in subsequent partitionings—but the centroids are adjusted both during and after each partitioning. For data point x in cluster i, if the centroid z_i is the nearest reference point, no adjustments are made and the algorithm proceeds to the next data point. However, if the centroid z_j of the cluster j is the reference point closest to data point x, then x is reassigned to cluster j, the centroids of the "losing" cluster i (minus point x) and the "gaining" cluster j (plus point x) are recomputed, and the reference points z_i and z_j are moved to their new centroids. After each step, every one of the k reference points is a centroid, or mean, hence the name "*k*-means." An example of clustering using the standard *k*-mean algorithm is shown in Figure 3.

There are a number of variants of the *k*-means algorithm. In some versions, the error measure E is evaluated at each step, and a data point is reassigned to a different cluster only if that reassignment decreases E. In MacQueen's original paper on the *k*-means method, the centroid update (assign data point to cluster, recompute the centroid, move the reference point to the centroid) is applied at each step in the initial partitioning, as well as during the iterations. In all of these cases, the standard *k*-means algorithm requires about the same amount of computation for a single pass through all the data points, or one iteration, as does Lloyd's algorithm. However, the *k*-means algorithm, because it constantly updates the clusters, is unlikely to require as many iterations as the less efficient Lloyd's algorithm and is therefore considerably faster.

The Continuous *k*-Means Algorithm

The continuous *k*-means algorithm is faster than the standard version and thus extends the size of the datasets that can be clustered. It differs from the standard version in how the initial reference points are chosen and how data points are selected for the updating process.

In the standard algorithm the initial reference points are chosen more or less arbitrarily. In the continuous algorithm reference points are chosen as a random sample from the whole population of data points. If the sample is sufficiently large, the distribution of these initial reference points should reflect the distribution of points in the entire set. If the whole set of points is densest in Region 7, for example, then the sample should also be densest in Region 7. When this process is applied to Landsat data, it effectively puts more cluster centroids (and the best color resolution) where there are more data points.

Another difference between the standard and continuous *k*-means algorithms is the way the data points are treated. During each complete iteration, the standard algorithm examines all the data points in sequence. In contrast, the continuous algorithm examines only a random sample of data points. If the dataset is very large and the sample is representative of the dataset, the algorithm should converge much more quickly than an algorithm that examines every point in sequence. In fact, the continuous algorithm adopts MacQueen's method of updating the centroids during the initial partitioning, when the data points are first assigned to clusters. Convergence is usually fast enough so that a second pass through the data points is not needed.

From a theoretical perspective, random sampling represents a return to MacQueen's original concept of the algorithm as a method of clustering data over a continuous space. In his formulation, the error measure E_i for each region R_i is given by

$$E_i = \int_{x \in R_i} \rho(x) \, \| x - z_i \|^2 dx \, ,$$

where $\rho(x)$ is the probability density function, a continuous function defined over the space, and the total error measure E is given by the sum of the E_i's. In MacQueen's concept of the algorithm, a very large set of discrete data points can be thought of as a large sample—and thus a good estimate—of the continuous probability density $\rho(x)$. It then becomes apparent that a random sample of the dataset can also be a good estimate of $\rho(x)$. Such a sample yields a representative set of cluster centroids and a reasonable estimate of the error measure without using all the points in the original dataset.

These modifications to the standard algorithm greatly accelerate the clustering process. Since both the reference points and the data points for the updates are chosen by random sampling, more reference points will be found in the densest regions of the dataset and the reference points will be updated by data points in the most critical regions. In addition, the initial reference points are already members of the dataset and, as such, require fewer updates. Therefore, even when applied to a large dataset, the algorithm normally converges to a solution after only a small fraction (10 to 15 percent) of the total points have been examined. This rapid convergence distinguishes the continuous *k*-means from less efficient algorithms. Clustering with the continuous *k*-means algorithm is about ten times faster than clustering with Lloyd's algorithm.

The computer time can be further reduced by making the individual steps in the algorithm more efficient. A substantial fraction of the computation time required by any of these clustering algorithms is typically spent in finding the reference point closest to a particular data point. In a "brute-force" method, the distances from a given data point to all of the reference points must be calculated and compared. More elegant methods of "point location" avoid much of this time-consuming process by reducing the number of reference points that must be considered—but some computational time must be spent to create data structures. Such structures range from particular orderings of reference points, to "trees" in which reference points are organized into categories. A tree structure allows one to eliminate entire categories of reference points from the distance calculations. The continuous *k*-means algorithm uses a tree method to cluster three-dimensional data, such as pixel colors on a video screen. When applied to seven-dimensional Landsat data, the algorithm uses single-axis boundarizing, which orders the reference points along the direction of maximum variation. In either method only a few points need be considered when calculating and comparing distances. The choice of a particular method will depend on the number of dimensions of the dataset.

Two features of the continuous *k*-means algorithm—convergence to a feasible group of reference points after very few updates and greatly reduced computer time per update—are highly desirable for any clustering algorithm. In fact, such features are crucial for consolidating and analyzing very large datasets such as those discussed in the accompanying article. □

Further Reading

James M. White, Vance Faber, and Jeffrey S. Saltzman. 1992. Digital color representation. U.S. Patent Number 5,130,701.

Stuart P. Lloyd. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory* IT-28: 129–137.

Edward Forgy. 1965. Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications, *Biometrics* 21: 768.

J. MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. Volume I, Statistics.* Edited by Lucien M. Le Cam and Jerzy Neyman. University of California Press.

Jerome H. Friedman, Forest Baskett, and Leonard J. Shustek. 1975. An algorithm for finding nearest neighbors. *IEEE Transactions on Computers* C-24: 1000-1006. [Single-axis boundarizing, dimensionality.]

Jerome H. Friedman Jon Louis Bentley, and Raphael Ari Finkel. 1977. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software* 3: 209–226. [Tree methods.]

Helmuth Späth. 1980. *Cluster Analysis Algorithms for Data Reduction and Classification of Objects.* Halsted Press.

Anil K. Jain and Richard C. Dubes. 1988. *Algorithms for Clustering Data*. Prentice Hall.

The biography of Vance Faber appears on page 149.