Histogram-Based Method for Effective Initialization of the K-Means Clustering Algorithm

Caroline Gingles and **M. Emre Celebi**
Department of Computer Science,
Louisiana State University in Shreveport,
Shreveport, LA, USA
ecelebi@lsus.edu

Abstract

K-means is undoubtedly the most widely used partitional clustering algorithm. Unfortunately, this algorithm is highly sensitive to the initial selection of the cluster centers. Numerous initialization methods have been proposed to address this drawback. Many of these methods, however, have superlinear complexity in the number of data points, which makes them impractical for large data sets. On the other hand, linear methods are often random and/or sensitive to the order in which the data points are processed. These methods are generally unreliable in that the quality of their results is unpredictable. In this paper, we propose a linear, deterministic, and order-invariant initialization method based on multidimensional histograms. Experiments on a diverse collection of data sets from the UCI Machine Learning Repository demonstrate the superiority of our method over the well-known maximin method.

1 Introduction

Clustering, the unsupervised classification of patterns into groups, is one of the most important tasks in exploratory data analysis (Jain, Murty, and Flynn 1999). Primary goals of clustering include gaining insight into (detecting anomalies, identifying salient features, etc.), classifying, and compressing data. Clustering has a long and rich history in a variety of scientific disciplines including anthropology, biology, medicine, psychology, statistics, mathematics, engineering, and computer science. As a result, numerous clustering algorithms have been proposed since the early 1950s (Jain 2010).

Clustering algorithms can be broadly classified into two groups: hierarchical and partitional (Jain 2010). Hierarchical algorithms recursively find nested clusters either in a top-down (divisive) or bottom-up (agglomerative) fashion. In contrast, partitional algorithms find all clusters simultaneously as a partition of the data and do not impose a hierarchical structure. Most hierarchical algorithms have quadratic or higher complexity in the number of data points (Jain, Murty, and Flynn 1999) and therefore are not suitable for large data sets, whereas partitional algorithms often have lower complexity.

Given a data set $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^D$, i.e., N vectors each with D attributes, hard partitional algorithms divide $\mathcal{X}$ into K exhaustive and mutually exclusive clusters $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_K\}$, $\bigcup_{i=1}^{K} \mathcal{P}_i = \mathcal{X}$, $\mathcal{P}_i \cap \mathcal{P}_j = \varnothing$ for $1 \leq i \neq j \leq K$. These algorithms usually generate clusters by optimizing a criterion function. The most intuitive and frequently used criterion function is the Sum of Squared Error (SSE) given by:

$$\text{SSE} = \sum_{i=1}^{K} \sum_{\mathbf{x}_j \in \mathcal{P}_i} \|\mathbf{x}_j - \mathbf{c}_i\|_2^2, \tag{1}$$

where $\|.\|_2$ denotes the Euclidean norm and $\mathbf{c}_i = 1/|\mathcal{P}_i| \sum_{\mathbf{x}_j \in \mathcal{P}_i} \mathbf{x}_j$ is the centroid of cluster $\mathcal{P}_i$ whose cardinality is $|\mathcal{P}_i|$.

The number of ways in which a set of N objects can be partitioned into K non-empty groups is given by Stirling numbers of the second kind:

$$\mathcal{S}(N, K) = \frac{1}{K!} \sum_{i=0}^{K} (-1)^{K-i} \binom{K}{i} i^N, \tag{2}$$

which can be approximated by $K^N/K!$ It can be seen that a complete enumeration of all possible clusterings to determine the global minimum of (1) is clearly computationally prohibitive. In fact, this non-convex optimization problem is proven to be NP-hard even for $K = 2$ (Aloise et al. 2009) or $D = 2$ (Mahajan, Nimbhorkar, and Varadarajan 2012). Consequently, various heuristics have been developed to provide approximate solutions to this problem (Tarsitano 2003). Among these heuristics, Lloyd's algorithm (Lloyd 1982), often referred to as the k-means algorithm, is the simplest and most commonly used one. This algorithm starts with K arbitrary centers, typically chosen uniformly at random from the data points. Each point is assigned to the nearest center and then each center is recalculated as the mean of all points assigned to it. These two steps are repeated until a predefined termination criterion is met. Algo. 1 shows the pseudocode of this procedure.

Because of its gradient descent formulation, k-means is highly sensitive to initialization. Adverse effects of improper initialization include empty clusters, slower convergence, and a higher chance of getting stuck in bad local minima (Celebi 2011). A large number of initialization methods have

been proposed in the literature (Celebi, Kingravi, and Vela 2013). Unfortunately, many of these have superlinear complexity in N, which makes them impractical for large data sets (note that k-means itself has linear complexity). In contrast, linear initialization methods are often random and/or order-dependent, which renders their results unrepeatable.

In this paper, we propose a histogram-based deterministic initialization method that is not only fast, but also effective. The rest of the paper is organized as follows. Section 2 describes the proposed initialization method. Section 3 provides the experimental results. Finally, Section 4 gives the conclusions and presents an outline of the future work.

input : $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^D$
output: $\mathcal{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_K\} \subset \mathbb{R}^D$
`Select a random subset` $\mathcal{C}$ `of` $\mathcal{X}$ `as the initial set of cluster centers`
while *termination criterion is not met* **do**
  `Assign each point to the nearest cluster`
  for $j \leftarrow 1$ **to** N **do**
    $i \leftarrow \underset{i \in \{1,\ldots,K\}}{\arg\min} \left\|\mathbf{x}_j - \mathbf{c}_i\right\|^2$
    $\mathcal{P}_i \leftarrow \mathcal{P}_i \cup \{\mathbf{x}_j\}$
  end
  `Recalculate the cluster centers`
  for $i \leftarrow 1$ **to** K **do**
    $\mathbf{c}_i \leftarrow \frac{1}{|\mathcal{P}_i|} \sum_{\mathbf{x}_j \in \mathcal{P}_i} \mathbf{x}_j$
  end
end

Algorithm 1: K-Means algorithm

2 Proposed Initialization Method

In this study, we approach the problem of determining an initial set of cluster centers from a density estimation perspective. Intuitively, the optimal centers are likely to be located in the densest regions of the probability density function. A histogram is the simplest and the most commonly used nonparametric density estimator (Scott 1992). To construct a histogram, each attribute of the data set is partitioned into a number of sub-intervals (a.k.a 'bins') and the number of data values that fall into each bin is calculated. Since the bins are of the same width, the most populated bins are also the areas of densest population, and likely to be good guesses for the initial cluster centers. The main issue in the implementation of a histogram-based density estimator is the determination of an appropriate bin width for each attribute. If the bin width is too small, the estimate becomes noisy, i.e., the bins suffer from significant statistical fluctuation due to the scarcity of samples. On the other hand, if the bin width is too large, the histogram cannot accurately represent the shape of the underlying distribution.

The most straightforward method for implementing a histogram-based representation involves mapping each histogram bin to an element of a D-dimensional integer array. Since the number of bins in a multidimensional histogram is the product of the number of bins in each dimension, this approach quickly becomes unmanageable as the number of attributes in a data set increases. Not only is it difficult to implement an array large enough to address a reasonable number of bins, but also iterating through such an array makes the algorithm very slow.

We circumvented the issue with the aforementioned naïve method by *not* addressing the bins in a multidimensional fashion. Instead, we proceed through the bins of the histogram one dimension (attribute) at a time, finding the most populated bin in each dimension. We use the centroid of the unidimensional bin found, that is, the arithmetic average of the projections of the points (that fall within this bin) on the current dimension, as the coordinate of the cluster center in the current dimension. Since a highly populated multidimensional bin[1] shares its population with each unidimensional bin that is a component of it, a highly populated unidimensional bin is likely to be a component of a highly populated multidimensional bin. This, however, does not solve the problem entirely. One might notice that, although a combination of highly populated unidimensional bins is far more likely to be a highly populated multidimensional bin than a randomly selected collection of unidimensional bins, there is information lost regarding which highly populated unidimensional bins in each dimension belong to the same highly populated multidimensional bin. A mismatch might give us a multidimensional bin that is nowhere near a cluster center, and, thus, is of little use. Rather than relying on guesses, we find the most populated bin in a dimension as described above except, rather than iterating to the next dimension, we make a temporary data set comprised only of the data points that were in the most populated bin. We then recursively pass this temporary data set to our algorithm again, this time considering the next dimension. This prevents the points in any of the nonselected unidimensional bins from being reconsidered in later passes, and thereby prevents any noise from other highly populated multidimensional bins. The algorithm continues in the aforementioned manner until there are no more dimensions, and then returns the centroids of the bins found at each step of the recursion.

It should be noted that, because the data points the algorithm is working with depend entirely on the results of the preceding step of the recursion, it is likely that the algorithm will return different results depending on the order in which the attributes are processed. For instance, a weakness of the algorithm lies in the fact that a unidimensional bin might contain more points than the unidimensional bin containing the most populated multidimensional bin. Because the algorithm finds the most populated unidimensional bin at each step of the recursion, it would, in this case, select the wrong unidimensional bin. Fortunately, in such cases, it is very likely that the algorithm would select the correct bin while searching for the next cluster center.

Fig. 1 illustrates this recursive algorithm on a two-dimensional toy data set. The algorithm first scans the unidimensional bins in the horizontal direction. The vertical ar-

[1]Note that a unidimensional bin is in fact a stack of multidimensional bins.

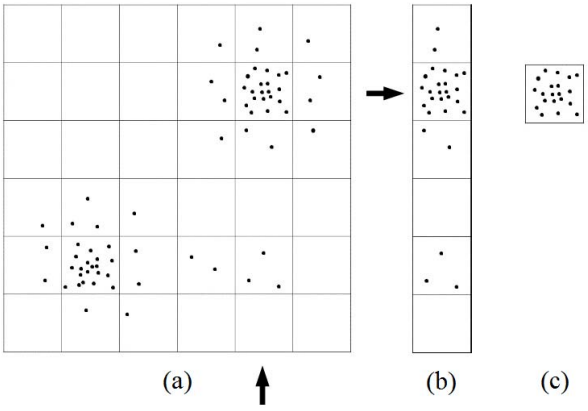

Figure 1: Illustration of the histogram traversal algorithm

row in panel (a) points to the most populated unidimensional bin found in this first pass. The algorithm then scans this unidimensional bin in the vertical direction. The horizontal arrow in panel (b) points to the most populated bin found in this second pass. Finally, the centroid of this bin, shown in panel (c), is taken as the first cluster center.

It should be noted that the aforementioned algorithm returns only one cluster center – that of the most populated bin in the data set. In order to determine K centers, we need execute this algorithm several times while preventing it from simply returning previously selected multidimensional bins. This can be accomplished by reducing the population of each unidimensional bin at each step of the recursion. The amount of reduction is determined based on the distance between the geometric center (midpoint) of each unidimensional bin and the centroids of the unidimensional bins corresponding to the previously selected centers. Let c be the centroid of a unidimensional bin chosen at a certain step. The population h^+ of a unidimensional bin with geometric center $\hat{c}$ is reduced as follows:

$$h^- = h^+ \left(1 - \exp\left(-kD(\hat{c} - c)^2\right)\right), \qquad (3)$$

where h^- is the reduced population of the unidimensional bin in question and k is a tuning parameter. Since the population of the previously selected bin is vastly reduced, this strongly steers the algorithm away from any previously selected unidimensional bins. This also makes the algorithm much less likely to pick bins very near to one another and therefore likely to be in the same cluster. Note that this population reduction is cumulative. In other words, every time the algorithm finds a cluster center, it reduces the population of all unidimensional bins by an amount inversely proportional to their distances to the selected unidimensional bin.

In this study, we used the following popular bin width estimation rules[2]: Scott (Scott 1979), Freedman–Diaconis (Freedman and Diaconis 1981), Silverman (Silverman 1986, p. 48), and Terrell (Terrell 1990). Table 1 gives the mathematical formulae for these rules. Here, n, q, and s denote

[2]The reader is referred to Sheather (Sheather 2004) and Heidenreich *et al.* (Heidenreich, Schindler, and Sperlich 2013) for recent surveys on density estimation.

Table 1: Bin Width Estimation Formulae

Rule	Formula	k
Scott	$3.49\, s\, n^{-1/3}$	8.52
Freedman–Diaconis	$2\, q\, n^{-1/3}$	2.11
Silverman	$0.9 \min\left\{s, \frac{q}{1.34}\right\} n^{-1/5}$	2.51
Terrell	$1.144\, s\, n^{-1/5}$	9.71

Table 2: Descriptions of the Data Sets (N: # points, D: # attributes, $\hat{K}$: # classes)

ID	Data Set	N	D	$\hat{K}$
1	Ecoli	336	7	8
2	Ionosphere	351	34	2
3	Iris (Bezdek)	150	4	3
4	Landsat Satellite (Statlog)	6,435	36	6
5	(Image) Segmentation	2,310	19	7
6	Shuttle (Statlog)	58,000	9	7
7	Vehicle Silhouettes (Statlog)	846	18	4
8	Wine Quality	6,497	11	7

the size, interquartile range[3], and standard deviation of the sample. The last column gives the empirically determined values for the tuning parameter used in Eq. (3). It should be noted that, on a larger collection of data sets, we have determined that the optimal range of k values for each bin width estimator is quite narrow.

Let $x_1, \ldots, x_N$ be the values of a particular attribute. The number of bins for this attribute is calculated as follows:

$$b = \left\lceil \frac{\max\{x_1, \ldots, x_N\} - \min\{x_1, \ldots, x_N\}}{w} \right\rceil, \qquad (4)$$

where w is the bin width estimate given by one of the aforementioned rules and $\lceil y \rceil$ is the ceiling function, which gives the smallest integer not less than y.

3 Experimental Results

The experiments were performed on eight commonly used data sets from the UCI Machine Learning Repository (Frank and Asuncion 2013). Table 2 gives the data set descriptions. For each data set, the number of clusters (K) was set equal to the number of classes ($\hat{K}$), as commonly seen in the related literature (Bradley and Fayyad 1998; Arthur and Vassilvitskii 2007; Su and Dy 2007; Celebi and Kingravi 2012; Celebi, Kingravi, and Vela 2013).

In clustering applications, normalization is a common preprocessing step that is necessary to prevent attributes with large ranges from dominating the distance calculations and also to avoid numerical instabilities in the computations. Two commonly used normalization schemes are linear scaling to unit range (min-max normalization) and linear scaling to unit variance (z-score normalization). In general, the former scheme is preferable to the latter since the latter is

[3]Interquartile range is defined as the difference between the upper and lower quartiles.

likely to eliminate valuable between-cluster variation (Milligan and Cooper 1988). For this reason, we used min-max normalization to map the attributes of each data set to the $[0, 1]$ interval.

The convergence of k-means was controlled by the disjunction of two criteria: the number of iterations reaches a maximum of 100 or the relative improvement in SSE between two consecutive iterations drops below a threshold (Linde, Buzo, and Gray 1980), i.e., $(\text{SSE}_{i-1} - \text{SSE}_i) / \text{SSE}_i \leq \epsilon$, where SSE_i is the SSE value at the end of the ith ($i \in \{1, \ldots, 100\}$) iteration. The convergence threshold was set to $\epsilon = 10^{-6}$.

The proposed histogram-based initialization method is compared to the commonly used maximin initialization method (Gonzalez 1985). This method chooses the first center $\mathbf{c}_1$ arbitrarily and the remaining $(K-1)$ centers are chosen successively as follows. In iteration i ($i \in \{2, \ldots, K\}$), the ith center $\mathbf{c}_i$ is chosen as the point with the greatest minimum ℓ_2 distance to the previously selected $(i-1)$ centers, i.e., $\mathbf{c}_1, \ldots, \mathbf{c}_{i-1}$. This method can be expressed in algorithmic notation as follows:

1. Choose the first center $\mathbf{c}_1$ arbitrarily from the data points.
2. Choose the next center $\mathbf{c}_i$ ($i \in \{2, \ldots, K\}$) as the point $\mathbf{x}_j$ that satisfies

$$\hat{j} = \underset{j \in \{1,\ldots,N\}}{\arg\max} \left(\min_{k \in \{1,\ldots,i-1\}} \|\mathbf{x}_j - \mathbf{c}_k\|_2^2 \right). \quad (5)$$

3. Repeat step #2 $(K-1)$ times.

Despite the fact that it was originally developed as a 2-approximation to the K-center clustering problem[4], maximin is commonly used as a k-means initializer. In this study, the first center is chosen as the centroid of $\mathcal{X}$ given by

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{j=1}^{N} \mathbf{x}_j \quad (6)$$

It is easy to see that $\mathbf{c}_1 = \bar{\mathbf{x}}$ gives the optimal SSE when $K = 1$.

It should be noted that, in this study, we do not attempt to compare a mixture of deterministic and random initialization methods. Instead, we focus on deterministic methods for two main reasons. First, these methods are generally computationally more efficient as they need to be executed only once. In contrast, random methods are inherently unreliable in that the quality of their results is unpredictable and thus it is common practice to perform multiple runs of such methods and take the output of the run that produces the least SSE. Second, several studies (Su and Dy 2007; Celebi and Kingravi 2012; Celebi, Kingravi, and Vela 2013) demonstrated that despite the fact that they are executed only once, some deterministic methods are highly competitive with well-known and effective random methods such as

[4]Given a set of N points in a metric space, the goal of *K-center* clustering is to find K representative points (centers) such that the maximum distance of a point to a center is minimized (Har-Peled 2011, p. 63). Given a minimization problem, a 2-*approximation* algorithm is one that finds a solution whose cost is at most twice the cost of the optimal solution.

Table 3: SSE Comparison of the Initialization Methods

		SSE	
ID	Method	Initial	Final
1	Maximin	47.9	19.3
	Scott	36.5	**18.5**
	Freedman–Diaconis	44.5	**18.5**
	Silverman	44.2	18.6
	Terrell	**32.1**	**18.5**
2	Maximin	**826.5**	826.5
	Scott	1139.9	**628.9**
	Freedman–Diaconis	913.1	**628.9**
	Silverman	865.3	**628.9**
	Terrell	986.3	**628.9**
3	Maximin	17.9	**7.0**
	Scott	**7.5**	7.1
	Freedman–Diaconis	14.5	**7.0**
	Silverman	10.5	**7.0**
	Terrell	13.5	7.1
4	Maximin	4815.9	**1741.6**
	Scott	4926.6	**1741.6**
	Freedman–Diaconis	**2841.8**	**1741.6**
	Silverman	3073.3	**1741.6**
	Terrell	5553.8	**1741.6**
5	Maximin	1084.5	433.3
	Scott	833.9	**387.0**
	Freedman–Diaconis	730.1	411.7
	Silverman	**636.8**	**387.0**
	Terrell	1045.4	414.7
6	Maximin	1817.8	725.8
	Scott	**792.1**	**235.0**
	Freedman–Diaconis	872.7	413.3
	Silverman	1048.1	413.5
	Terrell	1173.8	410.1
7	Maximin	466.0	237.7
	Scott	**320.7**	**223.5**
	Freedman–Diaconis	380.9	**223.5**
	Silverman	463.1	237.5
	Terrell	356.7	**223.5**
8	Maximin	732.6	399.4
	Scott	671.7	348.6
	Freedman–Diaconis	1085.8	**334.5**
	Silverman	822.4	337.2
	Terrell	**626.0**	346.8

Table 4: Rank Comparison of the Initialization Methods

	SSE Rank	
Method	Initial	Final
Maximin	4.00	4.63
Scott	**2.38**	2.44
Freedman–Diaconis	3.00	**2.06**
Silverman	2.63	2.81
Terrell	3.00	3.06

Bradley and Fayyad's method (Bradley and Fayyad 1998) and k-means++ (Arthur and Vassilvitskii 2007).

The performance of the initialization methods was quantified using two effectiveness (quality) criteria:

- ▷ **Initial SSE**: This is the SSE value calculated after the initialization phase, before the clustering phase. It gives us a measure of the effectiveness of an initialization method by itself.
- ▷ **Final SSE**: This is the SSE value calculated after the clustering phase. It gives us a measure of the effectiveness of an initialization method when its output is refined by k-means. Note that this is the objective function of the k-means algorithm, i.e., Eq. (1).

Table 3 gives the initial and final SSE values for the maximin method and the four variants (Scott, Freedman–Diaconis, Silverman, and Terrell) of the proposed histogram-based initialization method on the eight data sets. The best (lowest) SSE values are shown in **bold**. It can be seen that, with respect to initial SSE, variants of the histogram-based initialization method outperform the maximin method by a large margin, except on data set # 2. This means that in applications where an approximate clustering of the data set is desired, one of the variants of the proposed method can be used. There is significantly less variation among the initialization methods with respect to final SSE compared to initial SSE. In other words, the performance of the initialization methods is more homogeneous with respect to final SSE. This was expected because, being a local optimization procedure, k-means can take two disparate initial configurations to similar (or, in some cases, even identical) local minima. Nevertheless, as the table shows, except on data sets #3 and #4, variants of the histogram-based initialization method outperform the maximin method, in some cases (e.g., data set #6), by a large margin.

We also ranked the initialization methods based on their initial and final SSE values separately on each data set. In case of ties, the tied ranks were given the mean of the rank positions for which they are tied. For either effectiveness criterion, an ideal method would attain a rank of 1. Table 4 gives these ranks averaged over all data sets. As expected, there is *no* method that outperforms the rest consistently. However, the maximin method ranks second-to-last (4.00) and almost last (4.63) with respect to initial and final SSE, respectively. In contrast, the proposed method attains the best initial SSE rank (2.38) with Scott's rule and the best final SSE rank (2.06) with Freedman–Diaconis rule. Considering both effectiveness criteria simultaneously, the former rule appears to be slightly better than the latter one.

4 Conclusions and Future Work

In this paper, we presented a novel, histogram-based approach to initialize the k-means clustering algorithm. The method first normalizes the data set and then maps it to a multidimensional histogram, where the bin width in each dimension is calculated using a rule-of-thumb estimator. In order to determine the most populated K bins, this histogram is then traversed using an efficient recursive algorithm. Finally, the centroids of these bins are taken as the initial cluster centers. Besides being highly efficient, the proposed method is deterministic, which means that it needs to be executed only once prior to k-means. Experiments on a diverse collection of data sets from the UCI Machine Learning Repository demonstrated the superiority of our method over the popular maximin method. Future work includes testing the proposed method on a larger collection of data sets using additional performance criteria and more sophisticated bin width estimators.

5 Acknowledgments

This publication was made possible by a grant from the National Science Foundation (1117457).

References

Aloise, D.; Deshpande, A.; Hansen, P.; and Popat, P. 2009. NP-Hardness of Euclidean Sum-of-Squares Clustering. *Machine Learning* 75(2):245–248.

Arthur, D., and Vassilvitskii, S. 2007. K-Means++: The Advantages of Careful Seeding. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1027–1035.

Bradley, P. S., and Fayyad, U. 1998. Refining Initial Points for K-Means Clustering. In *Proceedings of the 15th International Conference on Machine Learning*, 91–99.

Celebi, M. E., and Kingravi, H. 2012. Deterministic Initialization of the K-Means Algorithm Using Hierarchical Clustering. *International Journal of Pattern Recognition and Artificial Intelligence* 26(7):1250018.

Celebi, M. E.; Kingravi, H.; and Vela, P. A. 2013. A Comparative Study of Efficient Initialization Methods for the K-Means Clustering Algorithm. *Expert Systems with Applications* 40(1):200–210.

Celebi, M. E. 2011. Improving the Performance of K-Means for Color Quantization. *Image and Vision Computing* 29(4):260–271.

Frank, A., and Asuncion, A. 2013. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml. University of California, Irvine, School of Information and Computer Sciences.

Freedman, D., and Diaconis, P. 1981. On the Histogram as a Density Estimator: L_2 Theory. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete* 57(4):453–476.

Gonzalez, T. 1985. Clustering to Minimize the Maximum Intercluster Distance. *Theoretical Computer Science* 38(2–3):293–306.

Har-Peled, S. 2011. *Geometric Approximation Algorithms*. American Mathematical Society.

Heidenreich, N. B.; Schindler, A.; and Sperlich, S. 2013. Bandwidth Selection for Kernel Density Estimation: A Review of Fully Automatic Selectors. *AStA Advances in Statistical Analysis* 97(4):403–433.

Jain, A. K.; Murty, M. N.; and Flynn, P. J. 1999. Data Clustering: A Review. *ACM Computing Surveys* 31(3):264–323.

Jain, A. K. 2010. Data Clustering: 50 Years Beyond K-means. *Pattern Recognition Letters* 31(8):651–666.

Linde, Y.; Buzo, A.; and Gray, R. 1980. An Algorithm for Vector Quantizer Design. *IEEE Transactions on Communications* 28(1):84–95.

Lloyd, S. 1982. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory* 28(2):129–136.

Mahajan, M.; Nimbhorkar, P.; and Varadarajan, K. 2012. The Planar k-Means Problem is NP-hard. *Theoretical Computer Science* 442:13–21.

Milligan, G., and Cooper, M. C. 1988. A Study of Standardization of Variables in Cluster Analysis. *Journal of Classification* 5(2):181–204.

Scott, D. W. 1979. On Optimal and Data-Based Histograms. *Biometrika* 66(3):605–610.

Scott, D. W. 1992. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons.

Sheather, S. J. 2004. Density Estimation. *Statistical Science* 19(4):588–597.

Silverman, B. W. 1986. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall/CRC.

Su, T., and Dy, J. G. 2007. In Search of Deterministic Methods for Initializing K-Means and Gaussian Mixture Clustering. *Intelligent Data Analysis* 11(4):319–338.

Tarsitano, A. 2003. A Computational Study of Several Relocation Methods for K-Means Algorithms. *Pattern Recognition* 36(12):2955–2966.

Terrell, G. R. 1990. The Maximal Smoothing Principle in Density Estimation. *Journal of the American Statistical Association* 85(410):470–477.