Hartigan's Method for K-modes Clustering and Its Advantages

Zhengrong Xiang[1] **Md Zahidul Islam**[2]

[1] College of Computer Science, Zhejiang University, China
Email: zolaxiang@gmail.com

[2] School of Computing and Mathematics, Charles Sturt University, Australia
Email: zislam@csu.edu.au

Abstract

Recently it has been shown that for k-means, Hartigan's method has better optimization performance than the prevalent Lloyd's method. Hartigan's method is a general idea of optimization heuristic. When considering moving a point to another cluster, it measures the exact change to the objective function. In this paper we develop a Hartigan's method for another important clustering objective: k-modes, which is popularly used for categorical data. The proposed algorithm is as efficient as the Lloyd's method on k-modes. Moreover, we rigorously prove that Hartigan's method can further improve some local optima achieved by Lloyd's method. Empirical evaluation verifies this conclusion. Furthermore, when these two methods are used independently, Hartigan's method also achieves better optimization performance.

Keywords: Cluster Analysis, K-modes, Lloyd's method, Hartigan's method

1 Introduction

Partitioning relocation clustering (Kaufman 2009, Everitt 2011) is an important and popular scheme for cluster analysis, typical example being k-means (Steinley 2006). K-modes (Huang 1998, Chaturvedi 2001) is a similar method as k-means, which is used specifically for categorical data. It defines the cluster center to be the "mode" of the attribute values of all records/objects in the cluster. The goal is also to minimize the sum of distances from objects to the respective modes. Like k-means, k-modes has a clear clustering objective, and it's efficient to achieve local optima. Since it was proposed, there has been a lot of related research, and has appeared in commercial products (Daylight 2010).

The optimization methods for k-means and k-modes are the same: iteratively assign objects to the nearest center (mean/mode), and update the centers. This method is called Lloyd's method (Lloyd 1982, MacQueen 1967). For k-means, there is actually a less well known method called Hartigan's method (Hartigan 1975, 1979). It proceeds one point at a time. For each point, move it to another cluster if the k-means objective is improved. Iteratively scan the data set until no more points can be moved. For Euclidean distance, there is a closed-form expression for deciding whether or not a move is profitable, so that Hartigan's method is as efficient as Lloyd's method.

Recently it has been shown that for k-means, Hartigan's method achieves better optimization performance than Lloyd's method (Telgarsky 2010). Specifically, the data partitions reached by Hartigan's method are tighter than the Voronoi partitions reached by Lloyd's method. They also discussed the characteristics of situations where Hartigan's method can improve on the final partitions of Lloyd's method. Empirical results support the above statements.

Lately another piece of research (Slonim 2013) further advocates the advantages of Hartigan's method over Lloyd's method for k-means. It developed a closed-form expression for any Bregman divergence, which Euclidean distance is an example of. Moreover, some types of problems are provided where Lloyd's method can not make any improvements while Hartigan's method can correctly converge.

In this paper, we focus on a similar issue concerning the quality of optimization algorithms, while we change the subject to k-modes. First note that Hartigan's method is a general idea of optimization heuristic. Unlike k-means, there is no previous research about how to use this idea on k-modes. The first part of our work develops an instantiation of Hartigan's method on k-modes, which has never been done to the best of our knowledge. The computational complexity of the proposed application of Hartigan's method on k-modes is as good as Lloyd's method on k-modes.

Secondly, we rigorously prove that when Lloyd's method gets stuck at a local optimum, sometimes it can be further optimized by Hartigan's method. Specifically, we characterize the kind of situations when the improvement is possible, and when it appears more often. We also prove that it's strictly not possible for Lloyd's method to improve upon Hartigan's method.

In the empirical evaluations, both synthetic and real-world data verify the above conclusion. Moreover, an interesting discovery is that Hartigan's method also outperforms Lloyd's method when the two methods are used independently. Finally, experiments also show that Hartigan's method requires less computations than Lloyd's method when the number of clusters and the number of features are large.

The second author would like to thank the Faculty of Business COMPACT Fund R4 P55 in Charles Sturt University, Australia.

2 Hartigan's Method For K-modes

2.1 K-modes and Lloyd's Method

First we briefly present the original k-modes. Suppose we have a data set including N objects X_i, $i = [1, N]$. The number of features/attributes is D, and we want K clusters: C_k, $k = [1, K]$. The objective function of k-modes is:

$$F = \sum_{k=1}^{K} \sum_{X_i \in C_k} d(X_i, M_k) \tag{1}$$

In the objective, M_k is the mode of cluster C_k, $d(X, Y)$ refers to the distance measure between two categorical objects. In k-modes, it's the simple matching distance (let x_j be the jth feature of X):

$$d(X, Y) = \sum_{j=1}^{D} d(x_j, y_j), \;\; d(x_j, y_j) = \begin{cases} 0; & if \; x_j = y_j \\ 1; & otherwise \end{cases} \tag{2}$$

The mode takes the most frequent value for each attribute, in order to reach minima. For example, if *red* is the most dominant value for the Color attribute, the mode takes *red* for this attribute.

Lloyd's method for k-modes is just like that of k-means: iteratively assign objects to the nearest mode, and update the modes. When no objects are reassigned, the algorithm converges to a local optimum.

2.2 Hartigan's Method

In this section, we propose Hartigan's method for k-modes, and show its efficiency. While Hartigan's method was used for k-means (Hartigan 1975, Telgarsky 2010), it has never been used in the past for k-modes.

Hartigan's method is a greedy heuristic for optimizing clustering objectives. Single data point is relocated to another cluster, if this move improves the objective (in our scenario, the value of the objective function decreases). Proceed until no points can be relocated, and the algorithm reaches to a local optimum.

To formalize: suppose an object t is under consideration. t is originally in cluster P. Moving t to another cluster Q has the following consequences: cluster P lost an object, the value of objective function for P also changes into F_{P-t}; cluster Q added an object, the objective becomes F_{Q+t}. If this relocation improves the total objective, then:

$$F_{P-t} + F_{Q+t} < F_P + F_Q \tag{3}$$

Rewritten as

$$\Delta F = (F_{P-t} - F_P) + (F_{Q+t} - F_Q) < 0 \tag{4}$$

Like in the work(Xiang 2013), we call this *transfer test*. For one object, there can be multiple destination clusters that satisfy the transfer test. Usually, picking the cluster that maximizes ΔF gives better overall performance (Tarsitano 2003).

Now we propose a technique to efficiently optimize the objective of k-modes with the basic idea of Hartigan's method. At the first sight, computing the transfer test requires computing values of the objective function, which is computationally expensive ($O(N^2)$). However, it can be carried out quite efficiently by computing only the changes: $F_{P-t} - F_P$, $F_{Q+t} - F_Q$.

Table 1: A Particular Frequency Table

Attributes	f_1		f_2		
Attribute Values	a	b	x	y	z
Frequency	7	3	4	4	2

Table 2: Computation of the Transfer Test

Departure Value	$F_{P-t} - F_P$
Not-Mode	-1
Is-Mode:	0
Is-Mode and Exists-Duplicated-Mode	-1

Arrival Value	$F_{Q+t} - F_Q$
Is-Mode	0
Not-Mode	1
Not-Mode and Is-Duplicated-Mode	0

For every cluster we need to keep a *frequency table* to record the frequencies of all attribute values. The table is stored in memory , and every entry should be accessed in $O(1)$. For example, say a cluster has 10 objects, with 2 features f_1 and f_2, then the table might be like the one shown in Table 1. Note that the mode is (a, x) or (a, y).

Because the objective is summed over different attributes independently, we only need to know how to compute the transfer test at a single attribute. First look at the change at the original cluster: $F_{P-t} - F_P$.

One possibility is that the value of object t is different from the value of mode. Say it's the value b in attribute f_1. The departure of t wouldn't change the mode, because the mode value is still the most frequent one. F_{P-t}, which is the sum of distances from the remaining objects to the mode, decreases by 1, because we lost *one* object.

The value of t can also be the same as the mode value. For example, the value a in attribute f_1, and x in attribute f_2. This can be further divided into two cases. One case is: if a departs, the mode doesn't change. Meanwhile, the objective value stays the same, because the departed object had a distance of 0.

The other case is when duplicated mode exists. In f_2, y is a duplicated mode for x. If x departs, the mode becomes y. The change is computed as follows: before the departure, the mode is x, and $F = \sum d(y, x) = 4$; after the departure, the mode is y, $F = \sum d(x, y) = 3$. So the result is objective value decreases by 1. Note that we don't need to check other objects who are neither x nor y, because their distances to the mode remain to be 1.

A similar analysis can be made for the arriving cluster Q, which we do not provide in detail. The results are summarized in Table 2.

In the above analysis, the changes of modes (or sometimes no change) have also been discussed. After each transfer, the modes need to be updated accordingly. This is a trivial task that can be done in O(D).

By referring to Table 2, the transfer test can be computed in O(D). Thus the algorithm has a complexity that is linear to the number of objects, which is a desired quality for partitioning clustering methods. The whole algorithm is as follows:

1. Initialize a partition (for example, a random one) of the data set.

2. Scan every object of the data set. If there is an object that satisfies the transfer test (computed by referring to Table 2), relocate it to the cluster that has the most decrease of the objective function, i.e. the one with the biggest ΔF. Update the modes and the frequency tables of the two involving clusters.

3. Repeat Step 2 until no objects are relocated in a full cycle scan of the whole data set.

3 Advantages of Hartigan's Method

In k-means, the prevalent optimization method is called Lloyd's method, which is also used by k-modes. Recently it has been shown that Hartigan's method is superior than Lloyd's method on k-means, in that better optima are discovered (Telgarsky 2010). In this section, we rigorously prove a similar result on k-modes.

Lloyd's method and Hartigan's method are two different heuristics that would end up with different optima. One question is, can the result of Lloyd's method be further improved by Hartigan's method? That is to say, if we use the result of Lloyd's method as the initial partition for Hartigan's method, are there any relocations that can be made? Or in the opposite direction, can the result of Hartigan's method be further improved by Lloyd's method?

Theorem 1. Any optimum achieved by Hartigan's method, can not be further improved by Lloyd's method.

Proof. Suppose there is an object t that should be relocated according to Lloyd's method, from cluster P to cluster Q. Without losing generality, we assume there is only one attribute. Let the attribute value of t also be t, the modes of cluster P and Q be m_P, m_Q.

According to Lloyd's method, $d(t, m_P) > d(t, m_Q)$. To satisfy this, there is only one possible combination: $d(t, m_P) = 1$, and $d(t, m_Q) = 0$.

Refer to Table 2. $d(t, m_P) = 1$ corresponds to a Not-Mode departure, with a change of -1. $d(t, m_Q) = 0$ corresponds to a Is-Mode arrival, with a change of 0. So the transfer test is satisfied, which means it's not a final result of Hartigan's method. The contradiction proves the theorem.

Theorem 2. For some optima achieved by Lloyd's method, they can be further improved by Hartigan's method. In other words, Hartigan's method can escape certain local optima which trap Lloyd's method.

Proof. To simplify, we also look at the case when there is only one attribute. In order to satisfy the transfer test, the change of departure has to be -1, and the change of arrival has to be 0. From Table 2, both departure and arrival have two cases being -1 and 0 respectively, thus four combinations:

1. Not-mode -> Is-mode: $(d(t, m_P) = 1) > (d(t, m_Q) = 0)$.

2. Not-mode -> Not-mode and Is-Duplicated-Mode: $(d(t, m_P) = 1) = (d(t, m_Q) = 1)$.

3. Is-mode and Exists-Duplicated-Mode -> Is-mode: $(d(t, m_P) = 0) = (d(t, m_Q) = 0)$.

4. Is-mode and Exists-Duplicated-Mode -> Not-mode and Is-Duplicated-Mode: $(d(t, m_P) = 0) < (d(t, m_Q) = 1)$.

We can see that for the first combination, the Lloyd's method will do the relocation. For the other three combinations, the Lloyd's method will do nothing and converges, while Hartigan's method will make the transfer. So if Lloyd's method ends up with a partition which satisfies any one of the three combinations, Hartigan's method can proceed, further improving the objective function.

Now let's see the implications from the two theorems. Firstly, how often is Hartigan's method helpful when it's used after Lloyd's method converges? Sometimes it's more likely for Lloyd's method to end up in either one of three cases listed above, thus it's more often that local optima found by Lloyd's method are improvable by Hartigan's method. What are the characteristics of such situations? One common characteristic of the three cases in the proof is that there are duplicated modes in clusters. Duplicated modes often appears in smaller clusters: say we have two attribute values a and b, it's a bigger probability for a cluster of 10 objects ending up splitting a and b (5 objects each) , than for a cluster of 1000 objects to make the split. So for a fixed data set, the more clusters we want to find, the more helpful is Hartigan's method. Also as the number of features rises, it's more likely there are duplicated modes on at least one of the features, thus Hartigan's method is more helpful for high dimensional data.

Secondly, after a single run of each of the two algorithms, Hartigan's method is not always better than Lloyd's method, even if they start with the same initial partition. This is because they take different greedy choices along the way, and usually end up in different parts of the optimization spaces.

Thirdly, although there seems to have no rigorous way to compare the performances of the two methods used independently, Hartigan's method should perform better on average under certain assumptions. Suppose we first run Lloyd's method and have a resulting partition Π_1. Then we run Hartigan's method, and somewhere in the process, it reaches the same partition Π_1. Then the final result, say Π_2, is definitely better than Π_1. This is the case when Hartigan's method outperforms Lloyd's method in a single run. If we make the assumption that in other cases, the two methods are head to head at the quality of local optima, then altogether Hartigan's method is better. This is validated in empirical evaluations.

4 Empirical Results

In this section, our goal is to show that Hartigan's method does outperform Lloyd's method. Specifically, applying Hartigan's method after Lloyd's method can show improvement of the objective; using two methods independently, Hartigan's method also achieves better averaged performance. With respect to computational cost, empirical results show that when the numbers of attributes and clusters grow, Hartigan's method requires less iterations than Lloyd's method.

When initializing the two methods, generally we can use either *random partition* or *random seeds*. Random partition is to assign all objects to random clusters; random seeds is to select K objects to be the initial modes, and assign the remaining objects to nearest modes. In our experiments, we found that the two initialization methods lead to similar performance comparisons of Hartigan's method and Lloyd's method. Thus we report only part of the results: for synthetic data, we report results from random parti-

Table 3: Improvement of using Hartigan's method after Lloyd's method: synthetic data sets

	Times	Max	Mean	Std
K=2	52	41	23.87	23.49
K=4	124	67	23.96	30.34
K=6	208	79	21.62	32.2
K=8	362	102	20.57	33.69
K=10	514	77	21.03	31.97
	Times	Max	Mean	Std
D=10	42	72	25.57	29.07
D=15	124	67	23.96	30.34
D=20	196	108	26.17	34.88
D=25	254	130	27.15	39.95
D=30	364	132	30.88	44.55

tion; for real-world data, we report results from random seeds. Also note that all the data are recorded by running algorithms for 1000 times.

4.1 Using Hartigan's method After Lloyd's Method

Here, we run Lloyd's method first, then on the resulting partition, we run Hartigan's method. Sometimes Hartigan's method can not proceed, but when it does proceed, the optima can be further improved.

First look at the results on synthetic data (categorical values are uniformly distributed) in Table 3. We vary the number of clusters K, and the number of features D, set the invariant parameters to be 1000 objects, 15 features and 4 clusters. On the columns, "Times" refers to the number of times (in 1000 runs) when Hartigan's method can proceed on the final results of Lloyd's method; "Max" and "Mean" refers to the maximum and average improvement of the objective function; "Std" refers to the standard deviation of the local optima achieved by Lloyd's method.

As we can see, the number of times when Hartigan's method is helpful increases when K and D increases, until it's quite significant (almost half of the times when K=10). This is consistent with our analysis after Theorem 2. Furthermore, from the comparison of maximum improvement, average improvement and the standard deviation, we can see that the improvements are quite significant, as they are comparable with the standard deviation.

For the real-world data, we picked three commonly used categorical data sets from UCI machine learning repository (Bache 2013): mushroom, soybean and congressional voting records. Except for the soybean data, which has only 35 records, we varied the number of clusters. The results are in Table 4. A similar trend from the synthetic data is found: as K increases, more often Hartigan's method further improves upon Lloyd's method. Although the average improvement is less significant than synthetic data (possibly due to attribute value distributions far from uniform), the maximum improvement is still quite large when comparing to the standard deviation.

4.2 Hartigan's Method and Lloyd's Method Independently

Here we run Hartigan's method and Lloyd's method independently and compare their performances. From the analysis after Theorem 2, we argue that generally Hartigan's method should outperform Lloyd's

Table 4: Improvement of using Hartigan's method after Lloyd's method: real-world data sets

data	K	Times	Max	Mean	Std
Mushroom	2	12	1270	137.25	1745.71
	4	315	2827	40.79	2657.93
	6	541	1415	78.11	1894.09
	8	628	6048	124.61	1625.3
Soybean	4	459	69	10.07	21.78
Congress	2	32	3	3.00	92.01
	4	119	110	13.45	37.75
	6	168	136	16.33	36.68
	8	321	104	14.49	36.33

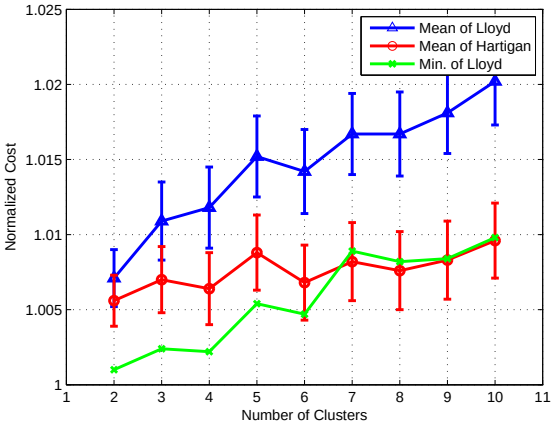

Figure 1: Normalized cost of two methods varying the number of clusters

method, because of the part played by Theorem 2. This is verified by our results here.

Figure 1 and 2 are results from synthetic data sets. Again the numbers of clusters and features are varied. The costs (value of objective function) are normalized by the best optima found by Hartigan's method. So only the minima of Lloyd's method are pictured. We can see that in all cases, Hartigan's method found better minimum optima, and better average optima. From the range of the error bars (standard deviation), we can conclude that the advantage is quite significant. Also the advantage grows as K or D increases. In some extreme cases, e.g, 7 clusters in Figure 1, even the minimum optimum by Lloyd's method is worse than the averaged optima by Hartigan's method.

In Table 5 for real-world data, we also listed the normalized costs of the two methods, their respective normalized standard deviations (Std), and the advantage of Hartigan's method over Lloyd's method measured by percentage (Per.). Like the results in Section 4.1, the advantage of Hartigan's method is less significant comparing to synthetic data, but it's there and it's more significant when K increases (see the decrease of advantage percentage).

4.3 Computational Cost

As we have shown, the Hartigan's method we developed for k-modes has the same time complexity with Lloyd's method for k-modes. However, the number of iterations required until converge is still a factor we are not sure about. Note that the equivalent notion of one iteration for Hartigan's method is one scan of the data set. Here we test the number of iterations

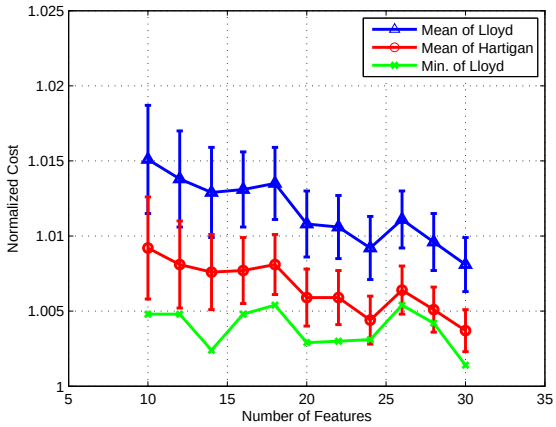

Figure 2: Normalized cost of two methods varying the number of features

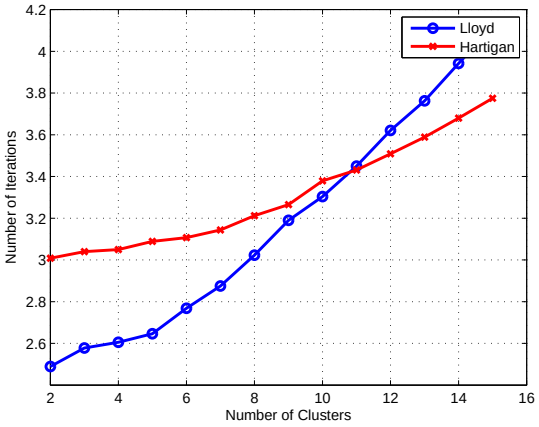

Figure 3: Iterations required by two methods varying the number of clusters

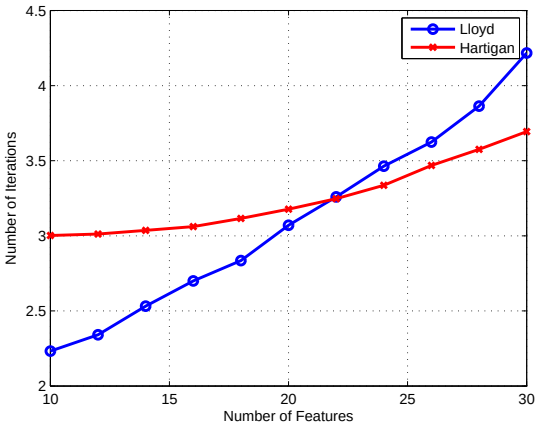

Figure 4: Iterations required by two methods varying the number of features

Table 5: Comparison of two methods independently used: real-world data sets

Data	k	Lloyd	Std	Hartigan	Std	Per.
Mush.	2	1.023	0.028	1.021	0.026	0.998
	4	1.054	0.055	1.050	0.053	0.996
	6	1.058	0.049	1.055	0.047	0.997
	8	1.072	0.044	1.067	0.038	0.995
Soy.	4	1.123	0.117	1.074	0.082	0.956
Con.	2	1.004	0.039	1.001	0.001	0.998
	4	1.060	0.026	1.058	0.024	0.997
	6	1.077	0.028	1.070	0.029	0.993
	8	1.088	0.030	1.079	0.030	0.992

for the two methods on the same synthetic data sets from above, by varying the number of clusters and the number of features, respectively. The unvaried parameters are 4 clusters and 15 features.

From Figure 3 and Figure 4, we can see that Hartigan's method begins to run less iterations when the number of clusters and features become larger. The crossovers are at 11 clusters and 22 features. The two numbers are relatively small, so we can say that generally Hartigan's method requires less computational cost.

As of why the number of iterations have this trend, a reasonable explanation is: as the number of clusters and features grow, Hartigan's method allows more relocations to happen in the early iterations. Since more work is done early on, less overall iterations are needed for Hartigan's method.

5 Final Remarks

To summarize, we developed an efficient Hartigan's method for k-modes, and proved that Hartigan's method can further improve the objective after Lloyd's method converges. Empirical results supported the conclusions. Independently used, Hartigan's method also achieves better optimization performance. Overall Hartigan's method is a better optimization method for k-modes.

Our result is consistent with the k-means literature (Telgarsky 2010, Slonim 2013), that Hartigan's method has better overall performance over Lloyd's method. The reason of this consistency is, although k-means and k-modes are two different objective functions, their Hartigan's methods have the same basic idea. The idea is to relocate objects according to the exact change of objective function. This works better for optimization than the more heuristic Lloyd's method.

For k-means (Telgarsky 2010, Slonim 2013), online k-means was also discussed as another optimization method, which has indistinguishable performance comparing with Lloyd's method. The Online method is slightly different in that it consider one point a time and update centers immediately after a relocation. For k-modes, our experiments also show close performances of the two methods, making it a trivial point that we didn't include.

References

Kaufman, L., Rousseeuw, P. J. (2009). Finding groups in data: an introduction to cluster analysis (Vol. 344). John Wiley & Sons.

Everitt, B. S., Landau, S., Leese, M. and Stahl, D. (2011). Cluster Analysis, 5th Edition. John Wiley & Sons.

Steinley, D. (2006). K-means clustering: a half-century synthesis. British Journal of Mathematical and Statistical Psychology, 59(1), 1-34.

Huang, Z. (1998). Extensions to the k-means algorithm for clustering large data sets with categorical values. Data mining and knowledge discovery, 2(3), 283-304.

Chaturvedi, A., Green, P. E., Caroll, J. D. (2001). K-modes clustering. Journal of Classification, 18(1), 35-55.

Daylight, Chemical Information Systems, Inc. http://www.daylight.com/

Lloyd, S. (1982). Least squares quantization in PCM. Information Theory, IEEE Transactions on, 28(2), 129-137.

MacQueen, J. (1967, June). Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability (Vol. 1, No. 14, pp. 281-297).

Hartigan, J. A. (1975). Clustering algorithms.

Hartigan, J. A., Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. Applied statistics, 100-108.

Telgarsky, M., Vattani, A. (2010). Hartigan's Method: k-means Clustering without Voronoi. In International Conference on Artificial Intelligence and Statistics (pp. 820-827).

Slonim, N., Aharoni, E., Crammer, K. (2013, August). Hartigan's K-means versus Lloyd's K-means: is it time for a change?. In Proceedings of the Twenty-Third international joint conference on Artificial Intelligence (pp. 1677-1684). AAAI Press.

Xiang, Z., Ji, L. (2013). The Use of Transfer Algorithm for Clustering Categorical Data. In Advanced Data Mining and Applications (pp. 59-70). Springer Berlin Heidelberg.

Tarsitano, A. (2003). A computational study of several relocation methods for k-means algorithms. Pattern recognition, 36(12), 2955-2966.

Bache, K., Lichman, M.: UCI Machine Learning Repository. (2013). University of California, School of Information and Computer Science, Irvine, http://archive.ics.uci.edu/ml