Speeding up k-means Clustering by Bootstrap Averaging

Ian Davidson and Ashwin Satyanarayana
Computer Science Dept, SUNY Albany, NY, USA, 12222.
{davidson, ashwin}@cs.albany.edu

Abstract

K-means clustering is one of the most popular clustering algorithms used in data mining. However, clustering is a time consuming task, particularly with the large data sets found in data mining. In this paper we show how bootstrap averaging with k-means can produce results comparable to clustering all of the data but in much less time. The approach of bootstrap (sampling with replacement) averaging consists of running k-means clustering to convergence on small bootstrap samples of the training data and averaging similar cluster centroids to obtain a single model. We show why our approach should take less computation time and empirically illustrate its benefits. We show that the performance of our approach is a monotonic function of the size of the bootstrap sample. However, knowing the size of the bootstrap sample that yields as good results as clustering the entire data set remains an open and important question.

1. Introduction and Motivation

Clustering is a popular data mining task [1] with k-means clustering being a common algorithm. However, since the algorithm is known to converge to local optima of its loss/objective function and is sensitive to initial starting positions [8] it is typically restarted from many initial starting positions. This results in a very time consuming process and many techniques are available to speed up the k-means clustering algorithm including preprocessing the data [2], parallelization [3] and intelligently setting the initial cluster positions [8].

In this paper we propose an alternative approach to speeding up k-means clustering known as bootstrap averaging. This approach is complimentary to other speed-up techniques such as parallelization. Our approach builds multiple models by creating small bootstrap samples of the training set and building a model from each, but rather than aggregating like bagging [4], we average similar cluster centers to produce a single model that contains k clusters. In this paper we shall focus on bootstrap samples that are smaller than the training data size. This produces results that are comparable with multiple random restarting of k-means clustering using all of the training data, but takes far less computation time. For example, when we take *T* bootstrap samples of size 25% of the training data set then the technique takes at least four times less computation time but yields as good as results if we had randomly restarted k-means *T* times using all of the training data. To test the effectiveness of bootstrap averaging, we apply clustering in two popular settings: finding representative clusters of the population and prediction.

Our approach yields a speedup for two reasons. Firstly, we are clustering less data and secondly because the k-means algorithm converges (using standard tests) more quickly for smaller data sets than larger data sets from the same source/population. It is important to note that we do not need to re-start our algorithm many times for each bootstrap sample.

Our approach is superficially similar to Bradley and Fayyad's initial point refinement (IPR) [8] approach that: 1) sub-samples the training data, 2) clusters each sub-sample, 3) clusters the resultant cluster centers many times to generate refined initial **starting** positions for k-means. However, we shall show there are key differences to their clever alternative to randomly choosing starting positions.

We begin this paper by introducing the k-means algorithm and explore its computational behavior. In particular we show and empirically demonstrate why clustering smaller sets of data leads to faster convergence than clustering larger sets of data from the same data source/population. We then introduce our bootstrap averaging algorithm after which we discuss our experimental methodology and results. We show that for bootstrap samples of less size than the original training data set our approach performs as well as standard techniques but in far less time. We then discuss the related Bradley and Fayyad technique IPR and discuss differences to our own work. Finally, we conclude and discuss future work.

2. Background to k-means Clustering

Consider a set of data containing n instances/observations each described by m attributes.

The k-means clustering problem is to divide the n instances into k clusters with the clusters partitioning the instances ($x_1 \ldots x_n$) into the subsets $Q_{1\ldots k}$. The subsets can be summarized as points ($C_{1\ldots k}$) in the m dimensional space, commonly known as centroids or cluster centers, whose co-ordinates are the average of all points belonging to the subset. We shall refer to this collection of centroids obtained from an application of the clustering algorithm as a model. K-means clustering can also be thought of as vector quantization with the aim being to minimize the vector quantization error (also known as the distortion) shown in equation (1). The mathematical trivial solution is to have a cluster for each instance but typically $k << n$.

$$VQ = \tfrac{1}{2}\sum_{i=1}^{n} D(x_i, C_{f(x_i)}), \text{where D is a distance function} \quad (1)$$
and f(x) returns the closet cluster index toinstance i

What is known today as the k-means clustering algorithm was postulated in a number of papers in the nineteen sixties [5][6]. The algorithm is extremely popular appearing in leading commercial data mining suites offered by SAS, SPSS, SGI, ANGOSS [7]. Typically the initial centroid locations are determined by assigning instances to a randomly chosen cluster though alternatives exist [8]. After initial cluster centroid placement the algorithm consists of the two following steps that are repeated until convergence. As the solution converged to is sensitive to the starting position the algorithm is typically restarted many times.

1) The assignment step: instances are placed in the closest cluster as defined by the distance function.

$$f(x_i) = \arg\min_j D(x_i, C_j)$$

2) The re-estimation step: the cluster centroids are recalculated from the instances assigned to the cluster.

$$C_j = \sum_{f(x_i)=j} \frac{x_i}{|Q_j|}$$

These two steps repeat until the re-estimation step leads to minimal changes in centroid values. Throughout this paper we use the version of the k-means clustering algorithm commonly found in data mining applications. We use the Euclidean distance, randomly assign instances to clusters to obtain initial cluster locations and normalize each attributes' value to be between 0 and 1 by dividing by the attributes' range. The k-means algorithm performs gradient descent on the distortion error surface and hence converges to a local optimum of its loss function (the distortion). Convergence can be measured in a number of ways such as the sum of the changes in the cluster centroids between adjacent iterations does not exceed some very small number epsilon (in this paper 10^{-5}).

3. Computational Complexity of k-means Clustering

As before, let n be the number of instances to cluster, m be the number of attributes/columns for each instance and k the number of clusters. Then it is well known that if the algorithm performs i iterations then the algorithm complexity is O($kmni$) [9]. While k, m and n are known before the algorithm begins execution, as stated earlier the algorithm is typically run until convergence occurs, meaning that, i is, at invocation, unknown. Of course i can be predetermined if the test for convergence is abandoned, but this is computationally inefficient.

For the rest of this section we derive results showing that i (number of iterations) is directly proportional to n (the number of instances) for a given data source and standard tests of convergence. That is, smaller data sets will converge in less iterations than larger data sets drawn from the same population. Without loss of generality, we assume a univariate data set and Euclidean distance. We first state the distortion which k-means tries to minimize, take the derivative of this expression with respect to the cluster centroid value, set to 0 and solve to derive the expression that minimizes the distortion. We find that this is as expected, the cluster centroid update expression as shown below.

$$VQ = \tfrac{1}{2}\sum_{k}^{K}\sum_{i=1}^{n}\delta(f(x_i), k).(C^t{}_k - x_i)^2$$
$$= \tfrac{1}{2}\sum_{k}^{K} VQ_k$$
$$VQ_k = \tfrac{1}{2}\sum_{i, f(x_i)=k}^{n}(C^t{}_k - x_i)^2$$
$$= \tfrac{1}{2}\sum_{i, f(x_i)=k}^{n}\left[(C^t{}_k)^2 - 2C^t{}_k x_i + {x_i}^2\right]$$
take first order derivative, set to zero and solve
$$\frac{\partial VQ_k}{\partial C^t{}_k} = \tfrac{1}{2}\sum_{i, f(x_i)=k}^{n}\left[2C^t{}_k - 2x_i\right]$$
$$0 = |Q_k| C^t{}_k - \sum_{i, f(x_i)=k}^{n} x_i,$$
$$C^t{}_k = \sum_{i, f(x_i)=k}^{n} x_i \Big/ |Q_k| \quad (2)$$

We now derive an expression to calculate the change in a cluster centroid between time *t-1* and *t* below:

$$\Delta C_k = C^t{}_k - C^{t-1}{}_k$$
$$= \frac{1}{|Q_k|} \sum_{i, f_t(x_i)=k}^{n} (x_i) - C^{t-1}{}_k$$
$$= \frac{1}{|Q_k|} \sum_{i, f_t(x_i)=k}^{n} \left(x_i - C^{t-1}{}_k\right) \quad (3)$$

as the summation occurs $|Q_k|$ *times.*

A similar analysis to our own has been performed while illustrating that k-means is performing Newton's gradient descent with a learning rate inversely related to the cluster size [10] as illustrated in equation (3). Furthermore, from equation (4) we see that the size of cluster centroid change is dependent on the change in the number of instances assigned to a cluster in adjacent iterations, that is the number of instances assigned to *k* at time *t* but not *t-1* plus those not assigned to *k* at time *t-1* but assigned at time *t*. For a given data source, the larger the data set the more likely the condition associated with the first summation in equation (4) will occur towards the end of the k-means run as our experiments will illustrate later in this section. Whether improved tests of convergence that consider the data set size may remove this phenomenon remains an open question.

$\forall i$, *if* $f_t(x_i) = k \Rightarrow f_{t-1}(x_i) = k$ *and*
if $f_{t-1}(x_i) = k \Rightarrow f_t(x_i) = k$ *then* $\Delta C_k = 0$,
therefore:

$$\Delta C_k = \frac{\sum_{i, f_t(x_i) \neq f_{t-1}(x_i), f_t(x_i)=k \text{ or } f_{t-1}(x_i)=k}^{n} (x_i)}{\left[\sum_{i, f_t(x_i) \neq f_{t-1}(x_i)}^{n} \delta(f_t(x_i), k) + \sum_{i, f_t(x_i) \neq f_{t-1}(x_i)}^{n} \delta(f_{t-1}(x_i), k) \right]} \quad (4)$$

where $\delta(a,b) = 1$, *when* $a = b$, *zero otherwise*

We now empirically illustrate our earlier claim that the number of iterations until convergence is related to data set size on a number of real world data sets as shown in Table 1, Table 2 and Table 3. These tables measure the average number of iterations until convergence occurs against increasing data set sizes. Convergence occurs if the change in (between adjacent iterations) clusters centroid positions when summed across all attributes and clusters is less than 10^{-5}. Other convergence tests were applied such as:

- the change in cluster centers is less than a percentage of the smallest distance between two clusters,
- that one centroid's changes is below epsilon or
- that no changes in cluster centers locations occurred, that is $\forall k, Q_k^t = Q_k^{t-1}$

but our findings did not differ significantly. We report average results for 100 experiments (random restarts). It is important to note that for a particular data set the 100 unique random starting positions are identical regardless of data set size. That is, we start the algorithm from the same 100 starting positions regardless of dataset size. Each smaller data set is a subset of the larger data sets.

Average No. Iterations	10%	25%	50%	75%	100%
Digit	7.23	8.06	8.82	8.94	9.05
Pima	5.06	7.49	6.21	6.84	7.55
Image	5.97	4.82	7.05	7.06	9.93

Table 1. The average number of iterations for k-means to converge for a variety of data sets. Note that k=2 and results are average over 100 experiments (random restarts).

Average No. Iterations	10%	25%	50%	75%	100%
Digit	8.51	11.04	12.11	13.15	14.19
Pima	6.29	8.88	12.83	15.09	15.61
Image	7.36	12.73	17.74	20.06	23.84

Table 2. The average number of iterations for k-means to converge for a variety of data sets. Note that k=4 and results are average over 100 experiments (random restarts).

Average No. Iterations	10%	25%	50%	75%	100%
Digit	8.41	12.61	17.91	20.22	23.55
Pima	6.75	10.03	16.82	17.51	20.01
Image	7.99	11.49	12.59	13.37	16.23

Table 3. The average number of iterations for k-means to converge for a variety of data sets. Note that k=6 and results are average over 100 experiments (random restarts).

Figure 1 diagrammatically illustrates our experiments showing that for larger values of *k* that the number of iterations monotonically increases as the data set size increases.

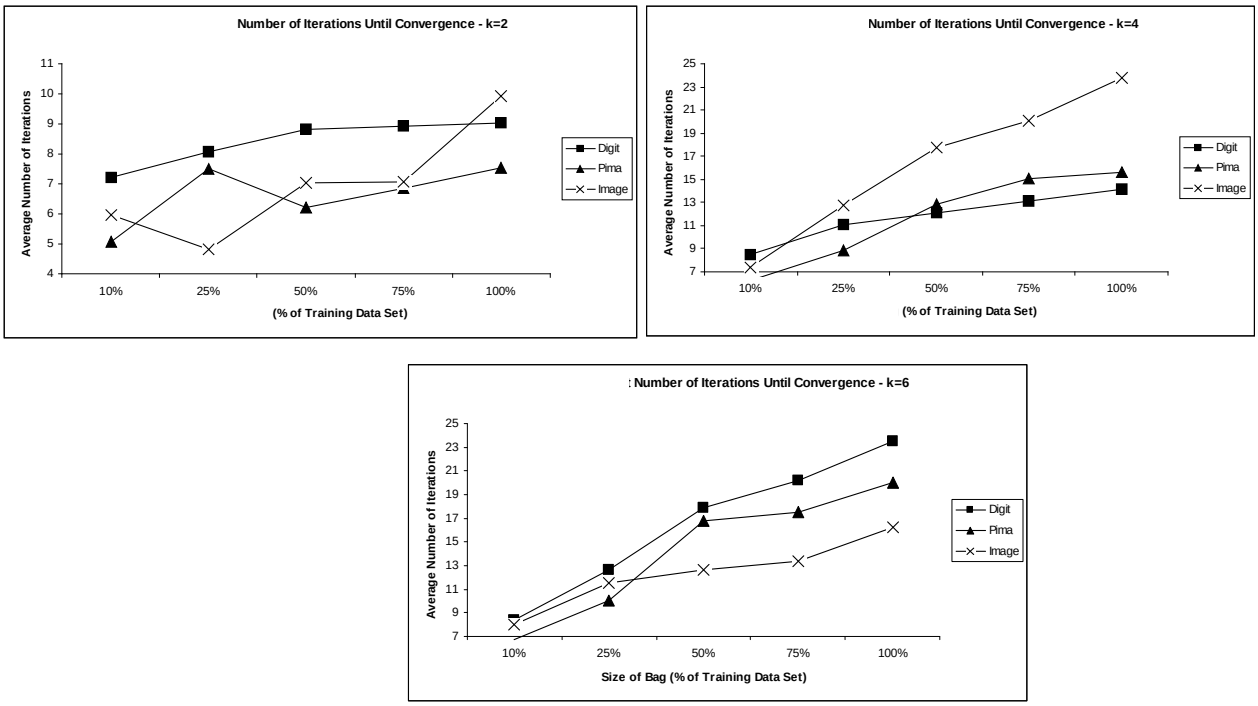

Figure 1. The average number of iterations for k-means to converge for a variety of data sets for k=2, k=4 and k=6. Results are average over 100 experiments (random restarts).

To illustrate the point that for a given source of data, clustering smaller data sets leads to faster convergence we map the trajectory the cluster algorithm takes through the instance space for different size data sets starting from the same starting position. To visualize these trajectories we will reduce the digit data set to two dimensions. This corresponds to the data set representing the starting position where the pen writing the digit was placed onto the tablet (an excellent predictor for the digit type). Some example trajectories are shown in Figure 2 and Figure 3.

In the 100 trials the average number of iterations for the 500 instance data set is 4.33 and for 2000 instances 9.12. Over the trials in only 6 trials was the number of iterations for the larger data set less than for the smaller data set. The figures illustrate that for both data sets the centroids quickly move to approximately the same location but the larger data set takes longer to converge. This is so because for larger data sets the condition associated with the first summation in equation (4) occurs more often towards the end of the k-means run (as seen in the right-hand figures above, it is more crowded near the final cluster centroid positions). This result is consistent with the results of Meek et' al [12] which found that when calculating a learning curve for mixture models that allowing the algorithm to reach convergence was not required, the results obtained after a few iterations was sufficient to determine the shape of the learning curve.

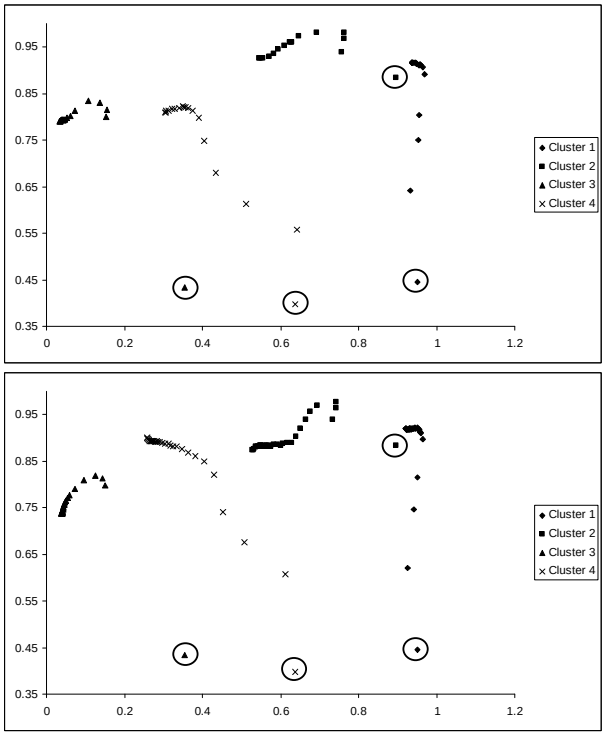

Figure 2. The trajectory of four cluster centroids through the instance space. The top figure is for 500 instances (17 iterations), the bottom for 2000 instances (31 iterations). Starting positions are circled.

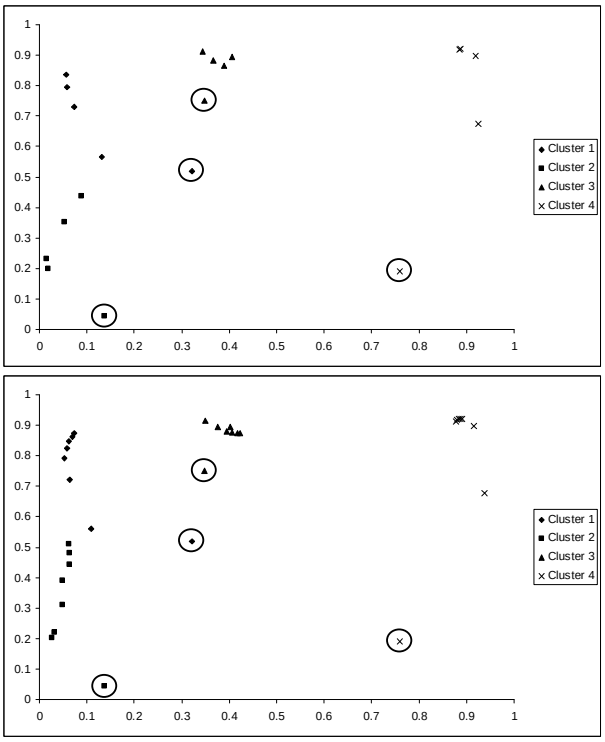

Figure 3. The trajectory of four cluster centroids through the instance space. The top figure is for 500 instances (5 iterations), the bottom for 2000 instances (8 iterations). Starting positions are circled.

4. Description of Our Approach

We now discuss our algorithm. Note only a single model is built from each bootstrap sample as k-means is only run once on each sample.

Algorithm: Bootstrap Averaging

Input: D:Training Data,T: Number of bags, K: Number of clusters
Output: A: The averaged centroids.

```
// Generate and cluster each bag
For i = 1 to T
      X_i = BootStrap(D)
      C_i = k-means-Cluster(X_i,K)
// Note C_i is the set of k cluster
// centroids and C_i = {c_i1, c_i2 … c_iK)
EndFor
// Group similar clusters into bins
// with the bin averages stored in B_1 …
// B_k their sizes are S_1 … S_k
For i = 1 to T
      For j = 1 to K
      Index = AssignToBin(c_ij)
//See section on signature based comparison
      B_Index += c_ij
      EndFor
EndFor
For i = 1 to K
      B_i /= S_i
      A_i = B_i
EndFor
```

Our approach involves averaging similar cluster centroids. We propose the following general-purpose method that is used throughout the paper, however, in practice problem specific approaches may be better.

4.1 Signature Based Cluster Grouping

For each cluster centroid we create a signature that can be generated quickly and group clusters according to the signature. We use the positions of the attributes and their values to create a signature of the form:

$Signature(c_{ij})=\Sigma_l\, c_{ijl}*2^l$, where c_{ijl} is the l^{th} attribute for the j^{th} cluster of the i^{th} model.

As each attribute is scaled to be between zero and one this creates a signature with the range 0 till 2^{m+1} as there are m attributes. After the signature from each cluster is derived we sort them in ascending order and divide them into k equally size intervals to form the groups. Throughout this paper we use this method. In the future we plan to explore the feasibility of using the approaches proposed by Strehl and Ghosh [11] that involves combining multiple partitions/clusters.

5. How Much Our Approach Speeds up Clustering

If we average T bootstrap samples of size $1/s$ of the training data then we expect our technique to obtain as good results as performing T random restarts but in less computation time. As described earlier we expect a speedup for two reasons. The speedup due to clustering less data will be of magnitude s since the relative complexity of the clustering process will be $O(TkmnI)$ versus $O(Tkmn/sI)$ assuming the same number of iterations until convergence. However, as discussed earlier for different sized data sets from the same population/source the time to convergence is not the same, typically $I_{n/s}<I_n$, this provides an even greater speed up which the next set of experiments quantify.

6. Experimental Results

The first use of our approach is its ability to find more accurate estimates of the generating mechanism. To test this ability we need to artificially create data to know the actual generating mechanism. We created an artificial data set of six clusters with six attributes. All attributes are Gaussians with a mean of zero and a standard deviation of 0.5 except the i^{th} attribute of the i^{th} cluster which has a mean of one and a standard deviation of 0.5. Formally, $C_{Gen}=\{c_1 \ldots c_6\}$, $c_i = \{c_{ij}=1$, if $i=j$, otherwise 0, $j = 1$ to $6\}$. We can measure the goodness of a clustering solution by its Euclidean

distance to these generating mechanisms. We generated 3000 instances from this distribution fifty times. For each sample we took 20 bootstrap samples, built a single model from each and averaged the cluster centroids to produce a single model. We group similar cluster centroids using the method described earlier.

The bootstrap averaging approach takes at least five times less computation time than clustering all of the data after performing 20 random restarts. If the number of random restarts is increased to 50 and even 100 the best model (minimum distortion) from the random restart approach still does not yield better results than bootstrap averaging. The distance to the true cluster centers is shown in Figure 4. This figure shows that for 10% bootstrap sample sizes that the averaged model is further away than the best model from random restarts. However, for 20% bootstrap sample size the averaged model is closer to the generating mechanism. Determining the precise size of the bootstrap sample when performance is as good as clustering the entire data set remains an open question.

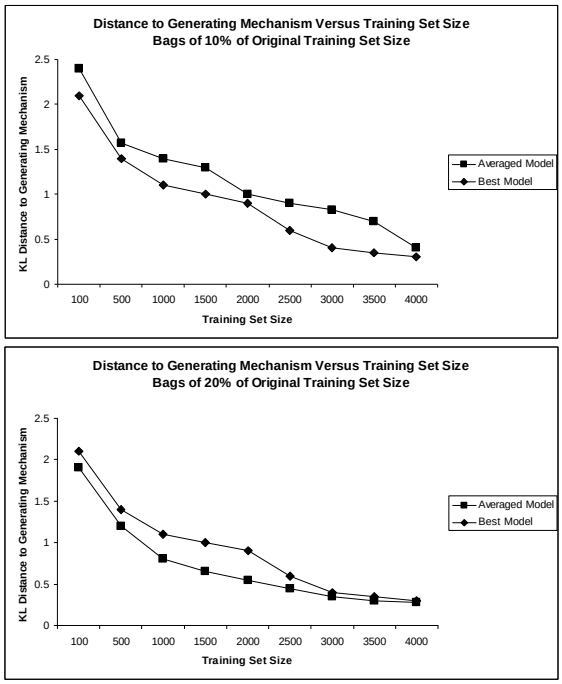

Figure 4. Training data size against mean distance (over 50 samples) to the generating mechanism for averaged model (20 bags) and best single model found from 20 random restarts for the entire data set. The top graph is for bootstrap samples of size 10% and the bottom graph 20% of the entire data set. The decrease in computation time by using the bootstrap averaging approach is never less than five times.

We now show results for the size of the bootstrap sample against predictive accuracy. Though prediction is not a common application of clustering, it allows us to show the performance of our approach on real world problem where we do not know the generating mechanism or true model. We find as before that averaging smaller bootstraps yields as accurate results as random restarts on all of the data but in less computation time. In all experiments we drew 50 random samples from the data, divided this data into a training set (70%) and test set (30%). For each data set, we randomly restarted k-means 50 times and selected the model that minimized the distortion. We compare this model against the model obtained by averaging over 50 bootstrap samples.

Digit Data Set: The accuracy of the averaged model and computation time for the digit data set (predicting 3 or 8, the most difficult digit prediction decision in this data set) for different sized bootstrap samples are shown in Figure 5. The best model found from all of the data after the random restarts has a mean accuracy of 31.70% and the total computation time is 22 CPU minutes compared to the averaged model's accuracy 30.8% found in approximately 5 CPU minutes when using 25% of data size bootstrap sample. Note that in all our results we report the user time reported by the Linux time command. This corresponds to the amount of CPU time that was dedicated to the process apart from system kernel calls which was negligible in all case (under 0.01seconds). We find that similar speedups hold for other data sets. However, the size of the reduced bags when the accuracy of the averaged model is the same as obtained from random restarts from all of the data varies between data sets.

Image and Letter Datasets: We see in Figure 6 and Figure 7 that for the Image and Letter data sets that the computation speedup is approximately a factor of 3.98 and 3.54 respectively. This is so as the bootstrap sample size must increase to **40%** to obtain an acceptable accuracy.

Determining the correct size of the reduce bag remains an open question and we hope to explore literature from the learning curve area [12] to address this question in our future work. Our results indicate that the size of the bootstrap sample size is a monotonic function of the averaged model's distance to the generating mechanism (true model). This is an advantage over the work by Bradley and Fayyad where the size of the sub-samples leads to indifferent results [section 3.2, 8].

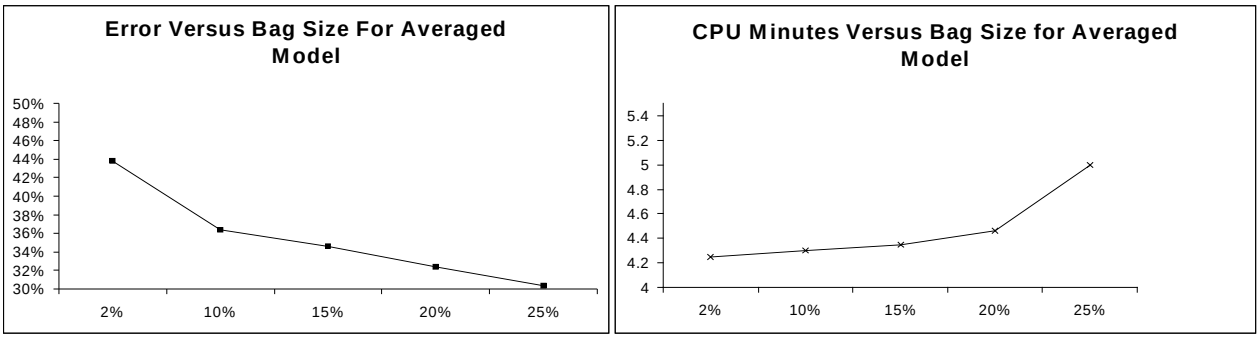

Figure 5. Digit Data Set. Training data size against predictive error (over 50 samples) (left graph) and computation time (right graph) for the averaged model. Note: Using the entire data set the computation time is 22 CPU minutes and accuracy is 31.7% as compared to 30.8% in approximately 5 CPU minutes when using 25% of data size bootstrap sample

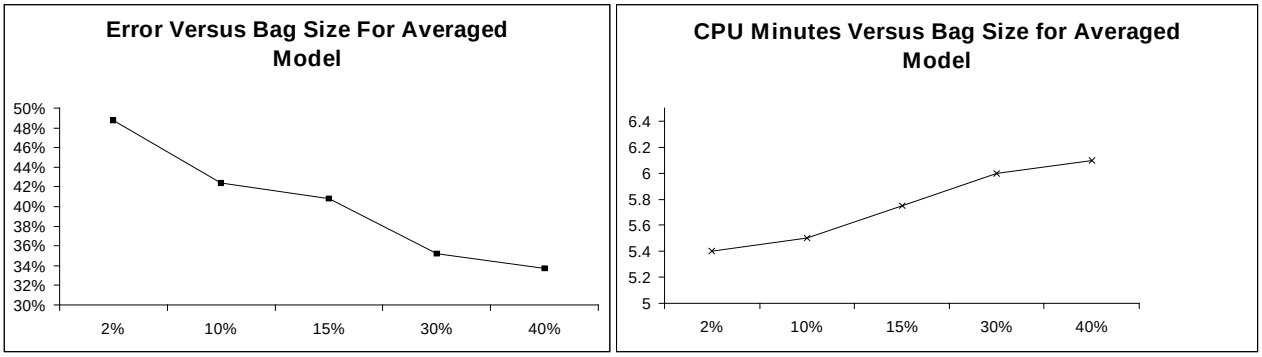

Figure 6. Image Data Set. Training data size against predictive error (over 50 samples) (left graph) and computation time (right graph) for the averaged model. Note: Using the entire data set the computation time is 23.5 CPU minutes and accuracy is 32.7%, as compared to accuracy of around 32.9% and 5 CPU minutes when using 40% of data size bootstrap sample.

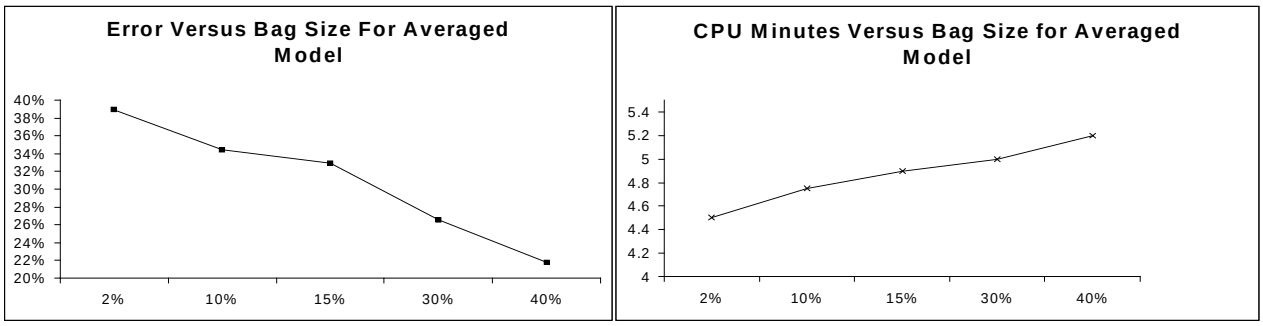

Figure 7. Letter Data Set. Training data size against predictive error (over 50 samples) (left graph) and computation time (right graph) for the averaged model. Note: Using the entire data set the computation time is 18.2 CPU minutes and accuracy is 20.6%, as compared to accuracy of around 21.0% and 5.2 CPU minutes when using 40% of data size bootstrap sample.

Table 4 shows a summary table of the situation when the bootstrap averaged accuracy equals the accuracy of clustering if all the data is used. This table shows the expected speed up ([1 / column 2] * [column 3 / column 4], see section 5 for details) and the actual speedup. The two numbers differ as the expected speed up does not consider the extra overhead such as the time required to generate the bootstrap samples. We did not attempt to optimize our code and hence expect the real difference between the expected and actual figures to be closer.

	Bootstrap Sample Size Required	Random Restarts: Ave. Number of Iterations	Bootstrap Averaging: Ave. Number of Iterations	Expected (Actual) Speed Up
Digit	25%	123.44	74.68	6.61 (4.63)
Image	40%	82.11	41.23	4.97 (3.98)
Letter	40%	72.57	43.45	4.17 (3.54)

Table 4. A summary of the statistics where the bootstrap averaged accuracy approximately equals the accuracy if all the data is clustered.

7. Discussion

We have shown that:

1. Bootstrap averaging *T* **subsets** of the data set will typically be more computationally efficient as randomly restarting the algorithm *T* times on the **entire data set**. (Section 3, Table 1, Table 2 and Table 3)
2. Bootstrap averaging on a proportion of the dataset can yield as accurate results as clustering the entire data set (see Figure 4): This produces results that are comparable with random restarting of k-means clustering using all of the training data, but takes far less computation time.
3. The results of bootstrap averaging monotonically improve as a function of the bootstrap sample size (see Figure 5, Figure 6 and Figure 7): As we increase the size of the bootstrap sample, the accuracy improves, until at some point the accuracy is comparable to that when the entire dataset is used.

Determining the size of the bags at which the averaged model performs as well as clustering the entire data set varies from data set to data set hence remains an open question.

A valid question is: how is our approach related to the IPR approach. For the rest of this section we empirically show that the two approaches obtain quite different results.

7.1 Is Bootstrap Averaging a Generalization of IPR

We begin by showing that the bootstrap averaging approach inherently does not produce a model that minimizes the distortion. That is, we are not compensating for some search inefficiency of the k-means algorithm as the IPR approach is effectively doing, instead we are minimizing another loss function as Table 5 indicates. The IPR approach attempts to find a good set of initial positions and then applies the k-means clustering algorithm to **further** minimize the distortion. The second column in this table is the result of applying bootstrap averaging. The first column was generated by initializing the clustering algorithm to the **averaged model** and clustering the entire training data. We find that the k-means algorithm that performs a gradient descent on the distortion error surface finds another model that further minimizes the distortion but yields worse performance. The averaged models are statistically significantly better at the 95% confidence level.

	Starting from Averaged Model	Averaged Model
Digit	27.9% (5.6)	23.6% (2.3)
Image	31.3% (4.6)	24.3% (4.1)
Pima	32.3% (3.2)	27.9% (3.3)
Letter	23.7% (4.1)	17.5% (2.3)
Abalone[1]	25.6% (5.1)	19.9% (3.1)
Adult	33.4% (7.1)	25.2% (4.5)

Table 5. Collection of data sets. The average and in parentheses standard deviation test set error statistics for the predictive ability of models found by starting k-means from the averaged model and the averaged model for 50 random divisions of training (70%) and test (30%) sets.

In our next set of experiments we show that bootstrap averaging performs quite differently to IPR. To illustrate this point clearly, we show that for bootstrap samples of equal size to the training data set that the results that k-means with IPR converges to is quite different from the bootstrap averaged model. In Table 6 the averaged model is significantly better (at the 95% confidence level) than k-means with IPR. The first column refers to the prediction ability of the model minimizing the distortion from 100 random restarts of the algorithm on the original training data. The second column refers to the prediction ability of the model minimizing distortion from 100 IPR selected starting solutions on the original training data. The final column (our approach) refers to the single model that is the average of all 20 bags.

[1] Predicting sex of abalone

	Random Restart	IPR Restart	Averaged Model
Digit[2]	30.6% (6.7)	29.5% (4.4)	23.6% (2.3)
Image	35.5% (9.8)	29.3% (5.2)	24.3% (4.1)
Pima	33.5% (4.5)	31.2% (3.1)	27.9% (3.3)
Letter	22.0% (6.4)	21.3% (3.2)	17.5% (2.3)
Abalone[1]	25.3% (7.9)	22.4% (4.5)	19.9% (3.1)
Adult	34.3% (10.3)	28.9% (5.4)	25.2% (4.5)

Table 6. Collection of data sets. The average and in parentheses standard deviation test set error statistics for the predictive ability of models found by applying k-means in a variety of situations over 50 random divisions of training (70%) and test (30%) sets.

8. Conclusion

K-Means clustering is a popular but time consuming algorithm used in data mining. It is time consuming as it converges to a local optimum of its loss function (the distortion) and the solution converged to is particularly sensitive to the initial starting positions. As a result its typical use in practice involves applying the algorithm from many randomly chosen initial starting positions.

In this paper we explore an approach we term bootstrap averaging. Bootstrap averaging builds multiple models by creating **small** bootstrap samples of the training set and building a **single** model from each, similar cluster centers are then averaged to produce a *single model* that contains *k* clusters. If we average *T* bags of size 1/*s* of the entire data set then our approach takes less time than randomly restarting the algorithm *T* times by a factor of at least *s*. Knowing the value of *s* where the averaged model performs as well as clustering the entire data set varies between data sets. The speedup is because the computational complexity of *k*-means is linear with respect to the number of data points. In practice we find that the speedup our approach provides is in fact greater than *s* since the standard test for convergence (the change in cluster centroids is less than some small number, epsilon) do not consider the size of the training data set. Our results indicate that the number of iterations of the algorithm until convergence is proportional to the size of the data set. In future work we will explore developing tests of convergence that factor in the data set size.

Our empirical results show that bootstrap sampling can achieve comparable results as clustering all of the data but in less computation time. We perform experiments to measure a clustering model's results in two ways: 1) distance to the generating mechanism and 2) predictive ability. However, knowing the size of the sample that performs as well as clustering the entire data set remains an open question. We hope to explore using the learning curve literature to determine potential ways to address this question.

Our research empirically shows that clustering bigger data sets is not always desirable. No doubt that clustering large data sets offer additional benefits namely producing better results, but our experiments indicate that bootstrapping smaller portions of the dataset can produce this benefit as well but at a reduction in computation time.

9. References

[1] Han J., and Kamber M., *Data Mining: Concepts and Techniques*, Morgan Kauffman, 2000.

[2] Pelleg, D. and Moore, A, "Accelerating Exact k-means Algorithms with Geometric Reasoning", KDD-99, Proc. of the Fifth ACM SIGKDD Intern. Conf. On Knowledge Discovery and Data Mining, page 277-281.

[3] Dhillon, I. S. and Modha, D. M., A Data Clustering Algorithm on Distributed Memory Multiprocessors, in *Large-Scale Parallel Data Mining, Lecture Notes in Artificial Intelligence, Volume 1759*, pages 245-260, 2000.

[4] L. Breiman. Bagging predictors. Machine Learning, 26(2):123-140, 1996.

[5] MacQueen, J., Some Methods for classification and analysis of multiattribute instances, Fiftieth Berkeley Symposium on Mathematics, Statistics and Probability, vol 1, 1967.

[6] Max, J., Quantizing for Minimum Distortion, IEEE Transactions on Information Theory, 6, pages 7-12, 1960

[7] H. Edelstein, The Two Crows Report: 1999. Available at http://www.twocrows.com/

[8] P. Bradley, U. Fayyad, Refining Initial Points for k-means Clustering. ICML 1998.

[9] Hartigan, J., *Clustering Algorithms*, Wiley Publishing, 1975.

[10] Bottou, L., and Bengio, Y., Convergence properties of the k-means algorithm. In G. Tesauro and D. Touretzky, editors, Adv. in Neural Info. Proc.

[2] Predicting digit 3 or 8

Systems, volume 7, pages 585--592. MIT Press, Cambridge MA, 1995.

[11] A. Strehl and J. Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal on Machine Learning Research (JMLR)*, 3:583-617, December 2002.

[12] C. Meek, B. Thiesson, and D. Heckerman. The learning-curve method applied to model-based clustering. Journal of Machine Learning Research, 2:397--418, 2002.