From: KDD-98 Proceedings. Copyright © 1998, AAAI (www.aaai.org). All rights reserved.

Scaling Clustering Algorithms to Large Databases

P.S. Bradley, Usama Fayyad, and Cory Reina

Microsoft Research
Redmond, WA 98052, USA
{bradley, fayyad, coryr}@microsoft.com

Abstract

Practical clustering algorithms require multiple data scans to achieve convergence. For large databases, these scans become prohibitively expensive. We present a scalable clustering framework applicable to a wide class of iterative clustering. We require at most one scan of the database. In this work, the framework is instantiated and numerically justified with the popular K-Means clustering algorithm. The method is based on identifying regions of the data that are compressible, regions that must be maintained in memory, and regions that are discardable. The algorithm operates within the confines of a limited memory buffer. Empirical results demonstrate that the scalable scheme outperforms a sampling-based approach. In our scheme, data resolution is preserved to the extent possible based upon the size of the allocated memory buffer and the fit of current clustering model to the data. The framework is naturally extended to update multiple clustering models simultaneously. We empirically evaluate on synthetic and publicly available data sets.

Introduction

Clustering is an important application area for many fields including data mining [FPSU96], statistical data analysis [KR89,BR93,FHS96], compression [ZRL97], vector quantization, and other business applications [B*96]. Clustering has been formulated in various ways in the machine learning [F87], pattern recognition [DH73,F90], optimization [BMS97,SI84], and statistics literature [KR89,BR93,B95,S92,S86]. The fundamental clustering problem is that of grouping together (clustering) similar data items.

The most general approach is to view clustering as a density estimation problem [S86, S92, BR93]. We assume that in addition to the observed variables for each data item, there is a hidden, unobserved variable indicating the "cluster membership". The data is assumed to arrive from a mixture model with hidden cluster identifiers. In general, a mixture model M having K clusters Ci, i=1,…,K, assigns a probability to a data point x: $\Pr(x \mid M) = \sum_{i=1}^{K} W_i \cdot \Pr(x \mid Ci, M)$ where Wi are the mixture weights. The problem is estimating the parameters of the individual Ci, assuming that the number of clusters K is known. The clustering optimization problem is that of finding parameters of the individual Ci which maximize the likelihood of the database given the mixture model. For general assumptions on the distributions for each of the K clusters, the EM algorithm [DLR77, CS96] is a popular technique for estimating the parameters. The assumptions addressed by the classic K-Means algorithm are: 1) each cluster can be effectively modeled by a spherical Gaussian distribution, 2) each data item is assigned to 1 cluster, 3) the mixture weights (Wi) are assumed equal. Note that K-Means [DH73, F90] is only defined over numeric (continuous-valued) data since the ability to compute the mean is required. A discrete version of K-Means exists and is sometimes referred to as harsh EM. The K-Mean algorithm finds a locally optimal solution to the problem of minimizing the sum of the L2 distance between each data point and its nearest cluster center ("distortion") [SI84], which is equivalent to a maximizing the likelihood given the assumptions listed.

There are various approaches to solving the optimization problem. The iterative refinement approaches, which include EM and K-Means, are the most effective. The basic algorithm is as follows: 1) Initialize the model parameters, producing a current model, 2) Decide memberships of the data items to clusters, assuming that the current model is correct, 3) Re-estimate the parameters of the current model assuming that the data memberships obtained in 2 are correct, producing new model, 4) If current model and new model are sufficiently close to each other, terminate, else go to 2).

K-Means parameterizes cluster Ci by the mean of all points in that cluster, hence the model update step 3) consists of computing the mean of the points assigned to a given cluster. The membership step 2) consists of assigning data points to the cluster with the nearest mean measured in the L2 metric.

We focus on the problem of clustering very large databases, those too large to be "loaded" in RAM. Hence the data scan at each iteration is extremely costly. We focus on the K-Means algorithm although the method can be extended to accommodate other algorithms [BFR98]. K-Means is a well-known algorithm, originally known as Forgy's method [F65,M67] and has been used extensively in pattern recognition [DH73, F90]. It is a standard technique used in a wide array of applications, even as a way to initialize more expensive EM clustering [B95,CS96,MH98,FRB98, BF98].

The framework we present in this paper satisfies the following *Data Mining Desiderata*:

1. Require one scan (or less) of the database if possible: a single data scan is considered costly, early termination if appropriate is highly desirable.
2. On-line "anytime" behavior: a "best" answer is always available, with status information on progress, expected remaining time, etc. provided
3. Suspendable, stoppable, resumable; incremental progress saved to resume a stopped job.

Copyright © 1998, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

4. Ability to incrementally incorporate additional data with existing models efficiently.
5. Work within confines of a given limited RAM buffer.
6. Utilize variety of possible scan modes: sequential, index, and sampling scans if available.
7. Ability to operate on forward-only cursor over a view of the database. This is necessary since the database view may be a result of an expensive join query, over a potentially distributed data warehouse, with much processing required to construct each row (case).

1 Clustering Large Databases

1.1 A Scalable Framework for Clustering

The scalable framework for clustering is based upon the notion that effective clustering solutions can be obtained by selectively storing "important" portions of the database and summarizing other portions. The size of an allowable pre-specified memory buffer determines the amount of summarizing and required internal book-keeping. We assume that an interface to the database allows the algorithm to load a requested number of data points. These can be obtained from a sequential scan, a random sampling (preferred), or any other means provided by the database engine. The process proceeds as follows:

1. Obtain next available (possibly random) sample from the DB filling free space in RAM buffer.
2. Update current model over contents of buffer.
3. Based on the updated model, classify the singleton data elements as:
   a. needs to be retained in the buffer (*retained set RS*)
   b. can be discarded with updates to the sufficient statistics (*discard set DS*)
   c. reduced via compression and summarized as sufficient statistics (*compression set CS*).
4. Determine if stopping criteria are satisfied: If so, terminate; else go to 1.

We illustrate this process in Figure 1. A basic insight is that not all data points in the data are of equal importance to the model being constructed. Some data points need to be used all the time (RS), and these can be thought of as similar to the notion of support vectors in classification and regression [V95]. Others are either discardable or reducible to a more efficient (hopefully equivalent) representation. While the framework of Figure 1 accommodates many clustering algorithms, we focus here on evaluating it for the K-means algorithm. Its applications to EM is given in [BFR98].

2 Components of the Scalable Architecture

The idea is to iterate over (random) samples of the database and "merge" information computed over previous samples with information computed from the current sample. Let n be the dimensionality of the data records. We are assuming the covariance matrix of each cluster is diagonal. Step 2 of the algorithm is basically a generalization of the classic K-Means algorithm which we call Extended K-means (Section 2.3) operating over data and sufficient statistics of previous/reduced data.For K-Means, a subset of data is represented by its mean and diagonal covariance matrix.

2.1 Data Compression

Primary data compression determines items to be discarded (discard set DS). Secondary data-compression takes place over data points not compressed in primary phase. Data compression refers to representing groups of points by their sufficient statistics and purging these points from RAM. The compressed representation constitutes the set CS. A group of singleton data elements deemed to be compressed will be called a *sub-cluster*. Finally, any remaining elements that defy primary and secondary compression are members of the retained set RS (singleton data points).

2.2 Sub-cluster Sufficient Statistics and Storage

Sub-clusters are compressed by locally fitting a Gaussian. Let $\{x^1, x^2, \ldots, x^N\} \subset R^n$ be a set of singleton points to be compressed. The sufficient statistics are the triple (SUM, SUMSQ, N), $SUM := \sum_{i=1}^{N} x^i \in R^n$, $(SUMSQ)_j := \sum_{i=1}^{N} \left(x_j^i\right)^2$, for $j=1,\ldots,n$. From the triple (SUM, SUMSQ, N) the computations of the sub-cluster mean $\bar{x}_{sub}$ and covariance diagonal s_{sub} are straightforward. For two sub-clusters i and j, having sufficient statistics $(SUM^i, SUMSQ^i, N^i)$ and $(SUM^j, SUMSQ^j, N^j)$, respectively, if merged will have: $(SUM^i + SUM^j, SUMSQ^i + SUMSQ^j, N^i + N^j)$ for their sufficient statistics. Let DS = {DS1, DS2,…, DSK } be a list of K elements where each element stores the sufficient statistics for the K sub-clusters determined in primary data-compression phase (Section 3). Similarly, let CS = { CS1, CS2, …,CSh } be the list of sufficient statistics triples for the h sub-clusters determined during the secondary data compression phase (Section 4).

2.3 Model Update over Sample + Sufficient Statistics

Step 2 of the scalable clustering algorithm outline consists of performing K-Means iterations over sufficient statistics of compressed, discarded, and retained points in RAM. In the Extended K-Means Algorithm, updates over singleton points are the same as the classic K-means algorithm updates. Updates over sufficient statistics involve treating

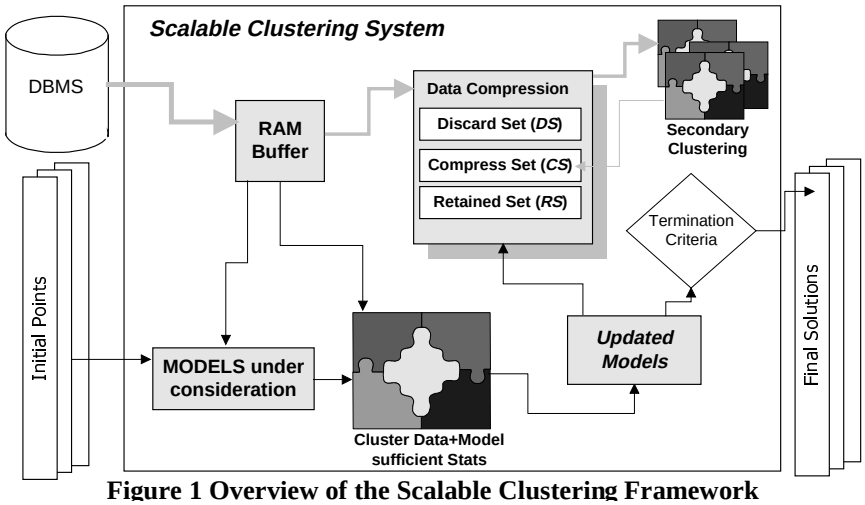


Figure 1 Overview of the Scalable Clustering Framework

each triplet (*SUM, SUMSQ, N*) as a data point with the weight of *N* items. The details are given in [BFR98]. Upon convergence of the Extended K-Means, if some number of clusters, say k < K have no members, then they are reset to the singleton data in the buffer that are furthest from their respective cluster centroids. Iterations are resumed until convergence over the RAM buffer contents.

3 Primary Data-Compression

Primary data compression is intended to discard data points unlikely to change cluster membership in future iterations. With each cluster j we associate the list DSj that summarizes the sufficient statistics for the data discarded from cluster j. Two methods for primary data compression exist. One is based upon thresholding the Mahalanobis radius [DH73] around an estimated cluster center and compressing all data points within that radius. The Mahalanobis distance from point x to the mean of a Gaussian $\bar{x}$ with covariance matrix S is: $D_M(x,\bar{x}) = \sqrt{(x-\bar{x})^T S^{-1}(x-\bar{x})} = \sqrt{\sum_{j=1}^{n} \frac{(x_j - \bar{x}_j)^2}{S_{jj}}}$, for a diagonal covariance matrix. The idea of first primary compression approach (PDC1) is to determine a Mahalanobis radius r which collapses p% of the newly sampled singleton data points assigned to cluster j. All data items within that radius are sent to the discard set DSj. The sufficient statistics for data points discarded by this method are merged with the DSj of points previously compressed in this phase on past data samples.

The second primary compression method (PDC2) creates a "worst case scenario" by perturbing the cluster means within computed confidence intervals. For each data point in the buffer, perturb the K estimated cluster means within their respective confidence intervals so that the resulting situation is a "worst case scenario" for a given singleton data point. Let the data point belong to cluster j. The perturbation consists of moving the mean of cluster j as far away from the data point as possible within its confidence interval and moving the mean of cluster $i \neq j$ as near to the data point as possible. If, after the resulting perturbation, the data point is still nearest to the perturbed center j, then it enters the set of points to be discarded DSj. This approach is motivated by the assumption that the mean is unlikely to move outside of the computed confidence interval. If the assignment of a given data point to a cluster remains fixed even if the current configuration of estimated cluster means change in a worst way, we assume that this assignment is robust over future data samples, and discard this point.

Confidence Interval Computations: To obtain a confidence interval for a mean in n-dimensions , n univariate confidence intervals are computed. The level of confidence of each of the n intervals is determined by exploiting the Bonferroni inequality [S84]. Let μ_j^l be the j-th component of the mean $\mu^l \in R^n$. We wish to determine $L_j^l < U_j^l$ such that $[L_j^l, U_j^l]$ is the univariate confidence interval on $\mu_j^l, j = 1,...,n$ such that the probability that all μ_j^l fall within their respective interval is greater than or equal to $1-\alpha$. Hence, we wish to satisfy: $P\left(\left(\mu_1^l \in \left[L_1^l, U_1^l\right]\right) \wedge \ldots \wedge \left(\mu_n^l \in \left[L_n^l, U_n^l\right]\right)\right) \geq 1-\alpha$. By the Bonferroni inequality, we have the following relationship: $P\left(\left(\mu_1^l \in \left[L_1^l, U_1^l\right]\right) \wedge \ldots \wedge \left(\mu_n^l \in \left[L_n^l, U_n^l\right]\right)\right) \geq 1 - \sum_{j=1}^{n} P\left(\mu_j^l \notin \left[L_j^l, U_j^l\right]\right)$

Hence it suffices that the univariate intervals are computed with confidence $\frac{\alpha}{n}$.

4 Secondary Data-Compression

The purpose of secondary data-compression is to identify "tight" sub-clusters of points amongst the data that we cannot discard in the primary phase. If singleton points designating a sub-cluster always change cluster assignment together, the cluster means can be updated exactly by the sufficient statistics of this sub-clusters. The length of the list CS varies depending on the density of points not compressed in the primary phase. Secondary data-compression has two fundamental parts: 1) locate candidate "dense" portions of the space not compressed in the primary phase, 2) applying a "tightness" (or "dense") criterion to these candidates. Candidate sub-clusters are determined by applying the vanilla K-means algorithm with a larger number of clusters k2 > K, initialized by randomly selecting k2 elements of the items in the buffer. Secondary data-compression takes place over all of the remaining singleton data (from all clusters and not for each individual cluster).

Once k2 sub-clusters are determined, the "tightness" criteria requiring all the sub-cluster covariances are bounded by the threshold β is applied. Consider one of the k2 sub-clusters containing $\tilde{N}$ elements and described by the triple: $\left(SUM, SUMSQ, \tilde{N}\right)$. Let $\bar{x}_{sub} = \frac{1}{\tilde{N}} \cdot SUM$, then if $\max_{j=1,\ldots,n} \left\{ \sqrt{\frac{1}{\tilde{N}} \cdot \left[\left(SUMSQ\right)_j - \tilde{N}\left(\bar{x}_{sub}\right)_j^2 \right]} \right\} \leq \beta$, the sub-cluster is "tight". Now suppose that k3 ≤ k2 sub-clusters pass this filter. These k3 sub-clusters are then merged with existing clusters in CS and with each other. This is performed by using hierarchical agglomerative clustering over the k3 + |CS| clusters. The nearest two sub-clusters are determined, and merged if their merge results in a new sub-cluster that does not violate the tolerance condition (see Section 2 for how the post-merge sufficient statistics are obtained), the merged sub-cluster is kept and the smaller ones removed.

5 One Scan, Many Solutions

The scheme presented involves updating a single model over a database. However, the machinery developed for compression and sufficient reduced representation also admits the possibility of updating multiple models simultaneously, within a single data scan.

K-means, as well as many other members of the iterative clustering algorithms, are well-known to be extremely sensitive to initial starting condition [DH73, F90]. In another paper, we study the initialization problem [BF98]. However, standard practice usually calls for trying multiple solutions from multiple random starting points. To support standard practice in clustering, we support the ability to explore multiple models. The key insights for this generalization are: 1) Retained points RS and the sets CS (representing local dense structures) are shared amongst the all models; 2) Each model, say Mi, will have its own discarded data sets DSMi (K sets, one for each cluster for

each model) -- if there are m models, there are a m×K discard sets; 3) The sufficient statistics for discarded data sets DSMi for one of the models Mi are simply viewed as members of the global CS by all models other than Mi.

Space requirements do not permit the presentation of the multiple model update. But the following provides insight. The overall architecture remains the same as the one shown in Figure 1, except that model updating and data compression steps are now performed over multiple models. Besides the 3 observations above, there is one data compression item worthy of further discussion: data discard order when multiple models are present. The algorithm decides on an individual data point basis which discard set fits it best. A data point that qualifies as a discard item for two models simply goes to the discard set of the model that it "fits" best. A data point cannot be allowed to enter more than one discard set else it will be accounted multiple times. Let x qualify as a discard item for both models M1 and M2. If it were admitted to both, then model M1 will "feel" the effect of this point twice: once in its own DS1 and another time when it updates over DS2 which will be treated as part of CS as far as M1 is concerned. Similarly for M2. By entering in one discard set, say DS1, the point x still affects M2 when M2 updates over CS (through DS1).

5.1 Complexity and Scalability Considerations

If, for a data set D, a clustering algorithm requires Iter(D) iterations to cluster it, then time complexity is |D| * Iter(D). A small subsample S $\subseteq$ D, where |S| << |D|, typically requires significantly fewer iteration to cluster. Empirically, it is reasonable to expect that Iter(S) < Iter(D). hence total time required to cluster n samples of size m is generally less than time required to cluster a single set of n×m data points.

The algorithm we presented easily scales to large databases. The only memory requirement is to hold a small sub-sample in RAM. All clustering (primary and secondary) occurs over the contents of the buffer. The approach can be run with a small RAM buffer and can effectively be applied to large-scale databases, as presented below. We have observed that running multiple solutions typically results in improved performance of the compression schemes since the synergy between the models being explored allow for added opportunity for compression. In turn this frees up more space in the buffer and allows the algorithm to maximize its exposure to new data during model updates.

We prefer to obtain a random sample from the database server. While this sounds simple, in reality this can be a challenging task. Unless one can guarantee that the records in a database are not ordered by some property, random sampling can be as expensive as scanning the entire database (using some scheme such as reservoir sampling, e.g. [J62]). In a database environment, a data view (i.e. the entity familiar to machine learning researchers) may not be materialized. It can be a result of query involving joins, groupings, and sorts. In many cases database operations impose special ordering on the result set, and "randomness" of the resulting database view cannot be assumed. In the case that random sampling is not available, it may be desirable to seed our scheme with true random samples, obtained by some other means, in the first few buffers.

6 Evaluation on Data Sets

The most straightforward way to "scale" an algorithm to a large database is to work on a sample of the data. Hence our comparisons will be mainly targeted at showing that our proposed scalable approach indeed performs better than simple random sampling. Clustering is particularly sensitive to choice of sample. Due to lack of space, we omit demonstration of this and more formal arguments for this increased sensitivity. See [BFR98] for more details. Our synthetic data sets (Section 6.1) are purposefully chosen so that they present a best-case scenario for a sampling-based approach. Demonstrating improvement over random sampling in this scenario is more convincing than scenarios where true data distributions are not known.

6.1 Synthetic Data Sets

Synthetic data was created for dimension d = 20, 50 and 100. For a given value of d, data was sampled from 5, 50, and 100 Gaussians (hence K=5 at d=20, K=d otherwise) with elements of their mean vectors (the true means) • were sampled from a uniform distribution on [-5,5]. Elements of the diagonal covariance matrices were sampled from a uniform distribution on [0.7,1.5]. Hence these are fairly well-separated Gaussians, an ideal situation for K-Means. A random weight vector W with K elements is determined such that the components sum to 1.0, then Wj*(total number of data points) are sampled from Gaussian j, j=1,…,K.

It is worthwhile emphasizing here the fact that data drawn from well-separated Gaussians in low dimensions are certainly a "best-case" scenario for the behavior of a random sampling based approach to scaling the K-means algorithm. Note that we are not introducing noise, and moreover we are providing the algorithm with the correct number of clusters K. In many real world data sets, the scenario is not "best-case", hence the behavior is likely worse (and indeed it is).

6.1.1 Experimental Methodology

The quality of an obtained clustering solution can be measured by how well it fits the data, or to measure the estimated means to the true Gaussian means generating the synthetic data. Measuring degree of fit of a set of clusters to a data set can be done using distortion1 or log-likelihood of the data given the model. Log-likelihood assumes the K-means model represents a mixture of K-gaussians with associated diagonal covariances and computes probability of data given the model. Comparing distance between one solution and another clustering solution requires the ability to "match up" the clusters in some optimal way prior to computing the distance between them. Suppose we want to measure the distance between a set of clusters obtained and the true Gaussian means. Let $\mu^l, l = 1,\ldots,K$ be the K true Gaussian means and let $\bar{x}^l, l = 1,\ldots,K$ be the K means estimated by some clustering algorithm. A "permutation" π is determined that minimizes: $\sum_{l=1}^{K} \left\| \mu^l - \bar{x}^{\pi(l)} \right\|_2$. The "score" is simply this quantity divided by K, giving the average distance between the true Gaussian means and a given set of clusters.

6.1.2 Results

The solutions we compare are the following: the solution provided by our scalable scheme (ScaleKM), the solution obtained by on-line K-means (OKM), and the solution obtained by the sampler K-means (sampKM) working on a sample equal to the size of the buffer given ScaleKM. Note

[1] Distortion is the sum of the L2 distances squared between the data items and the mean of their assigned cluster

Table 1. Results on Synthetic Data in 20 and 50 Dimensions

	20 D - 20K points, 5 clusters			50D - 50K points, 10 clusters		
Algorithm	D_{truth}	ratio D_{truth}	Δ likelihood	D_{truth}	ratio D_{truth}	Δ likelihood
OKM	187.0	30.4	-214390.3	73.1	6.4	-1673147.1
ScaleKM (10%)	9.3	1.5	-53134.2	**11.5**	**1.0**	**0**
ScaleKM (5%)	9.0	1.5	-54609.1	13.1	1.1	-31288.2
ScaleKM (1%)	**6.2**	**1.0**	**0.0**	15.8	1.4	-186094.2
SampKM(1%)	173.5	28.2	-166075.9	25.0	2.2	-249467.9
SampKM (10%)	173.5	28.2	-163766.5	17.3	1.5	-80536.5
SampKM (5%)	173.5	28.2	-163887.2	19.5	1.7	-129124.6

that the ultimate goal is to find a solution closer to the true generating means.

Our goal is to demonstrate that the method scales well as one increases dimensionality and number of clusters (parameters being estimated by clustering). We fist show results for the 20-D and 50-D data sets. Shown are distance to true solutions (Dtruth), ratios of Dtruth relative to best solutions (best are in bold), and change in log likelihood (Δ likelihood) as measured from best solution. These results represent averages over 10 randomly chosen starting points. The sampling solution is given a chance to sample 10 different random samples from the population. Hence random sampler results are over 100 total trials. ScaleKM is given a buffer size given as a percentage of size of data set.

It is worthy of note that results show scalable algorithm has robust behavior, even as its RAM buffer is limited to a very small sample. On-line K-means does not do well at all on these data sets. Measuring Distortion results in trends that follow Log likelihood.

Table 2. Results for 100-D Data.

Method	D_{truth}	ratio D_{truth}
OKM	324.7	1.8
ScaleKM (10%)	198.5	1.1
ScaleKM (5%)	197.2	1.1
ScaleKM (1%)	**181.6**	**1.0**
SampKM(1%)	261.0	1.4
SampKM (10%)	253.8	1.4
SampKM (5%)	255.9	1.4

Results on the 100-D data sets, with 100 dimensions and 100 clusters are given in Table 2.

6.2 Results on Real-World Data Sets

We present computational results on 2 publicly available "real-world" data sets. We are primarily interested in large databases. By large we mean hundreds of dimensions and tens of thousands to millions or records. It is for these data sets that our method exhibits the greatest value. We used a large publicly available data set available from Reuters News Service. We also wanted to demonstrate some of the results on Irvine machine Learning Repository data sets. For the majority of the data sets, we found that these data sets are too easy: they are low dimensional and have a very small number of records.

6.2.1 Experiments

The "quality" of the solution must be determined. Unlike the case of the synthetic data, we cannot measure distance to true solution here since "truth" is not known. However, we can use average class purity within each cluster as one measure of quality. The other measure, which is not dependent on a classification, is the distortion of the data given the clusters. Quality scoring methods are: distortion, log-likelihood, and information gain. The latter estimates the "amount of information" gained by clustering the database as measured by the reduction in class entropy. Recall that the class labels of the data are known, and an "ideal" clustering is I which clusters are "pure" in terms of the mix of classes appearing within them. This is scored by weighted entropy off the entire clustering:

$$Weighted \quad Entropy(K) \quad = \sum_{k=1}^{K} \left(\frac{Size(k)}{N} \right) ClusterEntropy(k).$$

Information Gain = Total Entropy – Weighted Entropy(K).

6.2.2 REV Digits Recognition Dataset

This dataset consists of 13711 data items in 64 dimensions. Each record represents the gray-scale level of an 8 x 8 image of a handwritten digit and each record is tagged as being in one of ten classes (one for each digit).

REV Results

The scalable K-Mean algorithm was run with a RAM buffer equivalent to 10% of REV dataset size. Comparisons are made with the Online K-Mean Algorithm [M67] (OKM) and with the K-Means algorithm operating over 10 random samples of size 10%. The results are averaged over 10 randomly chosen initial solutions (100 trials). We also ran the same experiments but using only 1% buffer sizes (1% samples). Results are shown in Table 3. One can also compare best solutions against best solutions. For the 1% buffer size, the best solution for SKM is 0.56 gain versus a best gain of 0.38 for the sampler K-means. For OKM, the best gain was 0.26. In the 10% case, even though the sampler did badly 90% of the time, it managed to find the best solution (0.57) once. The scalable method produced solutions, that are 2.2 times more "informative" than random sampling solutions, and 1.6 to 2 times better than on-line Kmeans. Note that our memory requirements can be driven really low and still maintain performance.

Table 3. Results for REV Digits Data Set.

Method	**Ave InfoGain**	**Std dev**
SKM(10% buffer)	0.320594	±0.018
sample10% x 10	0.1470409	±0.051
SKM-(1% buffer)	0.4015526	±0.138
Sample 1% x 10	0.1816977	±0.047
OKM	0.1957754	±0.083

6.2.3 Reuters Information Retrieval Dataset

The Reuters text classification database is derived from the original Reuters-21578 data set made publicly available as

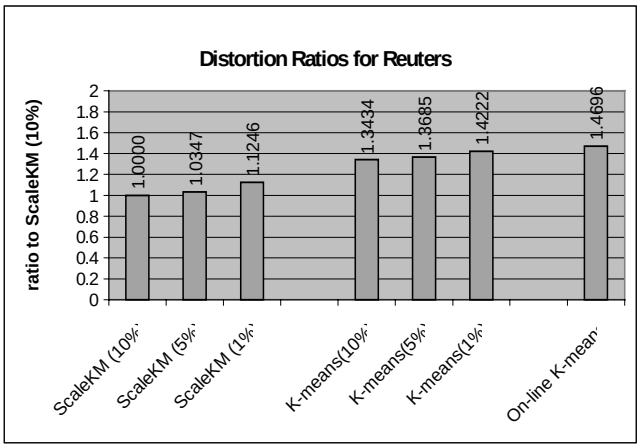
part of the Reuters Corpus, through Reuters, Inc., Carnegie Group and David Lewis (See: http://www.research.att.Com /~lewis /reuters21578 /README.txt for more details on this data set. This data consists of 12,902 documents. Each document is a news article about some topic: e.g. earnings, commodities, acquisitions, grain, copper, etc. There are 119 categories, which belong to some 25 higher level categories (there is a hierarchy on categories). The Reuters database consists of word counts for each of the 12,902 documents. There are hundreds of thousands of words, but for purposes of our experiments we selected the 302 most frequently occurring words, hence each instance has 302 dimensions indicating the integer number of times the corresponding word occurs in the given document. Each document in the IR-Reuters database has been classified into one or more categories. We use K=25 for clustering purposes to reflect the 25 top-level categories. The task is then to find the best clustering given K=25.

Reuters Results

For this data set, because clustering the entire database requires a large amount of time, we chose to only evaluate results over 5 randomly chosen starting conditions. Results are shown in Table 4. The chart shows a significant decrease in the total distortion measure. We show ratios of distortion as measured on the data. Note that on-line K-means in this case failed to find a good solution from any of the 5 initial points given it. We did manage to find some starting points (manually) that resulted in better clustering, but the comparison here must be made over exactly the same initial 5 points given the other methods. Distortion is roughly 30% better with the scalable approach than the corresponding random sampling based approach. One can also measure the degree of fit using the log-likelihood of the data given the model derived by K-means (25 Gaussians with diagonal covariances). If a model does not fit data well, log likelihood goes extremely negative (and overflows). This happened with on-line K-means and the sampling based approaches. The scalable runs produced finite likelihoods on all samples indicating that a better model was found.

Table 4. Results for Reuters Data.

Method	Log Likelihood
SKM-(10% buffer)	**-790.2K**
SKM-(5% buffer)	-871.3K
SKM-(1% buffer)	-903.8K
sample10%	-4803.0K
Sample5%	-35082.9K
Sample 1%	-81843.8K
OKM	-1908.2K



Since each document belongs to a category (there are 117 categories), we can also measure the quality of the achieved by any clustering by measuring the gain in information (reduction in cluster impurity) about the categories that each cluster gives (i.e. pure clusters are informative). The information gain for the scalable scheme was on average 13.47 times better than solution obtained by sampling approaches. If one compares best against best, we get that best scalable solution was 4.5 times better than the best random sampling solution. On-line K-means failed to converge on any good solutions (from all 5 starting points, it landed on solutions that put all the data in one cluster, hence did not improve impurity at all. Hence we cannot give a ratio. Again, given better (manually chosen) starting points, on-line Kmeans can be made to converge on better solutions.

7 Related Work

Since K-Means has historically been applied to small data sets, the state of the practice appears to be to try out various random starting points. Traditionally, K-Means is used to initialize more expensive algorithms such as EM [B95]. In fact, other methods to initialize EM have been used, including hierarchical agglomerative clustering (HAC) [DH73, R92] to set the initial points. Hence our choice of comparing against the random starting points approach. However, regardless of where a starting point comes from, be it prior knowledge or some other initialization scheme, our method can be used as an effective and efficient scalable means to proceed to a solution.

In statistics, all schemes we are aware of appear to be memory-based. A book dedicated to the topic of clustering large data sets [KR89] presents algorithm CLARA for clustering "large databases". However the algorithm is limited to 3500 cases maximum [KR89, p. 126]. The only options available in this literature to scale to large databases are random sampling and on-line K-means [M67]. We compare against both these methods in Section 6. On-line K-means essentially works with a memory buffer of one case. As we show in the results section, this methods does not compare well with other alternatives.

Within the data mining literature, the most relevant approach is BIRCH [ZRL97]. Other scalable clustering schemes include CLARANS [NH94] and DBSCAN [EKX95]. The latter two are targeted at clustering spatial data primarily. In [ZRL97] they compare against these schemes and demonstrate higher efficiency. The fundamental difference between BIRCH and the method proposed here are: 1) the data-compression step is performed prior to and independent of the clustering; 2) there is no notion of an allocated memory buffer in BIRCH, hence depending on data memory usage can grow significantly 3) BIRCH requires at least 2 scans of the data to perform clustering; 4) statistics maintained represent a simpler local model (strictly spherical Gaussians) with many more models built than our scheme requires. Our extended notion of 3 classes of data: RS, CS, and DS can be viewed as a generalization of BIRCH's all-discard strategy. We were not able to perform a comparative study with BIRCH, although this is certainly our plan. In BIRCH the expensive step is the maintenance and updating of the CF-tree. In our case the updates are much simpler. However,

our scheme requires a secondary clustering step which IRCH does not have. Again, secondary clustering in our case is limited to a small sample of the data. While we suspect our total run times will be lower than BIRCH's due to the fact that we require one or less data scans instead of two, and we do not have to maintain the large CF-tree structure. However, this claim needs to be supported by an empirical evaluation on similar machines and the same data sets.

References

[BR93] J. Banfield and A. Raftery, "Model-based gaussian and non-Gaussian Clustering", Biometrics, vol. 49: 803-821, pp. 15-34, 1993.

[B95] C. Bishop, 1995. Neural Networks for Pattern Recognition. Oxford University Press.

[B*96] R. Brachman, T. Khabaza, W. Kloesgen, G. Piatetsky-Shapiro, and E. Simoudis, "Industrial Applications of Data Mining and Knowledge Discovery." Communications of ACM 39(11). 1996.

[BMS97] P. S. Bradley, O. L. Mangasarian, and W. N. Street. 1997. "Clustering via Concave Minimization", in Advances in Neural Information Processing Systems 9, M. C. Mozer, M. I. Jordan, and T. Petsche (Eds.) pp 368-374, MIT Press, 1997.

[BF98] P. Bradley and U. Fayyad, "Refining Initial Points for K-Means Clustering", Proc. 15th International Conf on Machine Learning, Morgan kaufmann, 1998.

[BFR98] P. Bradley, U. Fayyad, and C. Reina, "Scaling Clustering Algorithms to Large Databases", Microsoft Research Report: May. 1998.

[CS96] P. Cheeseman and J. Stutz, "Bayesian Classification (AutoClass): Theory and Results", in in Advances in Knowledge Discovery and Data Mining, Fayyad, U., G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy(Eds.), pp. 153-180. MIT Press, 1996.

[DLR77] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data via theEM algorithm". Journal of the Royal statistical Society, Series B, 39(1): 1-38, 1977.

[DH73] R.O. Duda and P.E. Hart, Pattern Classification and Scene Analysis. New York: John Wiley and Sons. 1973

[EKX95] M. Ester, H. Kriegel, X. Xu, "A database interface for clustering in large spatial databases", Proc. First International Conferecne on Knowledge Discovery and Data Mining KDD-95, AAAI Press, 1995.

[FHS96] U. Fayyad, D. Haussler, and P. Stolorz. "Mining Science Data." Communications of the ACM 39(11), 1996.

[FPSU96] Fayyad, U., G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (Eds.) Advances in Knowledge Discovery and Data Mining. MIT Press, 1996.

[FDW97] U. Fayyad, S.G. Djorgovski and N. Weir, "Application of Classification and Clustering to Sky Survey Cataloging and Analysis", Computing Science and Statistics, vol. 29(2), E. Wegman and S. Azen (Eds.), pp. 178-186, Fairfax, VA: Interface Foundation of North America, 1997.

[FRB98] U. Fayyad, Cory Reina, and Paul Bradley, "Refining Initialization of Clustering Algorithms", *Proc. 4th international Conf. On Knowledge Discovery and Data Mining*, AAAI Press, 1998.

[F87] D. Fisher. "Knowledge Acquisition via Incremental Conceptual Clustering". Machine Learning, 2:139-172, 1987.

[F65] E. Forgy, "Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications", Biometrics 21:768. 1965.

[F90] K. Fukunaga, Introduction to Statistical Pattern Recognition, San Diego, CA: Academic Press, 1990.

[G97] C. Glymour, D. Madigan, D. Pregibon, and P. Smyth. 1997. "Statistical Themes and Lessons for Data Mining", Data Mining and Knowledge Discovery, vol. 1, no. 1.

[J62] Jones, "A note on sampling from a tape file". Communications of the ACM, vol 5, 1962.

[KR89] L. Kaufman and P. Rousseeuw, 1989. Finding Groups in Data, New York: John Wiley and Sons.

[M67] J. MacQueen, "Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. Volume I, Statistics, L. M. Le Cam and J. Neyman (Eds.). University of California Press, 1967.

[MH98] M. Meila and D. Heckerman, 1998. "An experimental comparison of several clustering methods", Microsoft Research Technical Report MSR-TR-98-06, Redmond, WA.

[NH94] R. Ng and J. Han, "Efficient and effective clustering methods for spatial data mining", Proc of VLDB-94, 1994.

[PE96] D. Pregibon and J. Elder, "A statistical perspective on knowledge discovery in databases", in Advances in Knowledge Discovery and Data Mining, Fayyad, U., G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy(Eds.), pp. 83-116. MIT Press, 1996.

[R92] E. Rasmussen, "Clustering Algorithms", in Information Retrieval Data Structures and Algorithms, W. Frakes and R. Baeza-Yates (Eds.), pp. 419-442, Upper Saddle River, NJ: Prentice Hall, 1992.

[S84] G.A.F. Seber, Multivariate Observations, New York: John Wiley and Sons, 1984.

[S92] D. W. Scott, Multivariate Density Estimation, New York: Wiley. 1992

[SI84] S. Z. Selim and M. A. Ismail, "K-Means-Type Algorithms: A Generalized Convergence Theorem and Characterization of Local Optimality." IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-6, No. 1, 1984.

[S86] B.W. Silverman, Density Estimation for Statistics and Data Analysis, London: Chapman & Hall, 1986.

[ZRL97] T. Zhang, R. Ramakrishnan, and M. Livny. "BIRCH: A new data clustering algorithm and its applications." Data Mining and Knowledge Discovery 1(2). 1997.