Weighted K-Means for Density-Biased Clustering

Kittisak Kerdprasop[1], Nittaya Kerdprasop[1], and Pairote Sattayatham[2]

[1] Data Engineering and Knowledge Discovery Research Unit,
School of Computer Engineering, Suranaree University of Technology,
111 University Avenue, Nakhon Ratchasima 30000, THAILAND
{kerdpras,nittaya}@ccs.sut.ac.th
http://www.sut.ac.th/engineering/computer/faculty/nittaya
[2] School of Mathematics, Suranaree University of Technology
111 University Avenue, Nakhon Ratchasima 30000, THAILAND
pairote@ccs.sut.ac.th

Abstract. Clustering is a task of grouping data based on similarity. A popular k-means algorithm groups data by firstly assigning all data points to the closest clusters, then determining the cluster means. The algorithm repeats these two steps until it has converged. We propose a variation called weighted k-means to improve the clustering scalability. To speed up the clustering process, we develop the reservoir-biased sampling as an efficient data reduction technique since it performs a single scan over a data set. Our algorithm has been designed to group data of mixture models. We present an experimental evaluation of the proposed method.

1 Introduction

Clustering is the automatic grouping of data based on similarity. There exists a large number of clustering techniques, but the most classical and popular one is the k-means algorithm [1]. Given a data set containing *n* objects, k-means partitions these objects into *k* groups. Each group is represented by the centroid of the cluster. Once cluster representatives are selected, data objects are assigned to the nearest centers. The algorithm iteratively selects new better representatives and reassigns data objects until no change is made. At this point the algorithm is said to converge. Even though k-means is an effective clustering algorithm, it can sometimes converge to a local optimum. Many methods [2,3,4,5] have been developed to extend the k-means with the common objective of avoiding converging to a bad local optimum. Some methods [6,7,8] search for the best initialization because k-means is known to be sensitive to initial point selection. Other research [9] seeks for the global optimum, at the cost of computation. These researches try to solve the problem of sub-optimal clustering and estimation the appropriate number of clusters [10,11].

Another difficulty of clustering with k-means is that it fails to identify clusters with large variation in sizes since original large clusters tend to be split. Clustering algorithms, such as DBSCAN [12] and CURE [13], have been developed to overcome this kind of difficulty. DBSCAN associates a data point with its density obtained by counting the number of points in a region of radius ε. The algorithm discovers clusters

A Min Tjoa and J. Trujillo (Eds.): DaWaK 2005, LNCS 3589, pp. 488-497, 2005.

by connecting regions with sufficient high density, a *MinPts* threshold. DBSCAN works well in spatial clustering, but it is sensitive to the selection of ε and *MinPts* and it fails to efficiently discover clusters with highly different densities. CURE algorithm represents a cluster by a set of points, instead of a single representative. Once the representative points are chosen, the algorithm then shrinks these points toward the centroid of the cluster according to a shrinking factor. CURE is an iterative hierarchical-based clustering that works well with discovering cluster of different sizes, but it is sensitive to the selection of representatives and shrinking factor. Moreover, with very large data set, these algorithms degrade considerably.

When clustering massive data set, data reduction is an effective technique to speed up the algorithm. Sampling [14,15,16] is a powerful data reduction paradigm to remedy the inherent complexity of clustering. Uniform random sampling in which every data point has the same probability of being selected has been used extensively in data mining and databases [17,18,19,20]. In the case of data sets with large variation in cluster sizes, density biased sampling [21,22,23] tends to be a better scheme. In density biased sampling, the probability that a data point will be included in the sample is varied by the density of a cluster.

Recent researches [21,22,23] propose several techniques to density biased sampling. Our work also follows this path with a step further on extending the k-means algorithm to work with a weighted sample. We propose an algorithm on density biased sampling based on the reservoir technique and a weighted k-means algorithm to cluster a data sample augmented with weights. The proposed algorithms are explained in Sections 2 and 3, respectively. We present the experimental results in Section 4. The conclusion and our future work are discussed in Section 5.

2 Data Reduction Biased by Density

On scalable popular and successful clustering methods such as k-means to work against large data sets, many algorithms like BIRCH [24] and CLARANS [14] employ the sampling technique to minimize data sets. In BIRCH, a CF-tree structure is built after an initial random sampling step. The CF-tree is used as a summarized data structure with statistical representations of space regions stored on leaf nodes. After the phase of CF-tree building, any clustering algorithm can be applied to the leaf nodes. CLARANS also uses uniform sampling to derive initial representative objects for the clusters.

The sampling technique used in these algorithms is uniform random sampling, which assigns every object the same probability of being included in the sample. But many data sets in real life do not follow the uniform distribution scheme. It instead seems to follow the Zipf’s distribution [25], for instance, income and population distribution. In these data sets, some areas such as large metropolitan area have much higher population density than the small cities. If all the populations have equal opportunity of being selected as a representative, sparse areas may be missed and not be included in the sample.

2.1 Density-Biased Sampling

Density biased sampling [21] is a sampling technique that takes into account the different sizes of the groups. Small groups or sparse regions are assigned higher probability to be included in the sample than the large groups or dense regions. By biasing the sampling process, small clusters will not be missed or overlooked as outliers.

Recent advancement on clustering very large data sets in which summarized data structure is even too big to fit into main memory, sampling is independently applied to the data set prior to the subsequent clustering phase. Palmer and Faloutsos [21] develop a non-uniform sampling method for clusters that differ very much in size and density. Their method is a generalization of uniform random sampling in that every group of data sets can be assigned different probability of being drawn. When sampling is biased by group density, smaller groups are oversampling, whereas larger groups are under-sampling. Since clusters are not known a priori, Palmer and Faloutsos combine the phase of density information extraction with the biased sampling phase using the hash-based approach. They argue that the inherent collision problem of any hash-based approach will not dramatically degrade the sample.

Nevertheless, their method is significantly affected by noise due to the tendency of oversampling noisy area. Our approach adopts the reservoir technique to eliminate the collision problem of hash-based approach and it is independent on the assumption regarding cluster distribution to avoid the impact of noise.

2.2 Density-Biased Reservoir Sampling

We propose a novel approach of adapting reservoir technique [26,27] to perform a density biased sampling on large data sets. Our algorithm can obtain a desired sample through a single data set scan. The proposed method is simpler and requires less resource than the hash-based method [21].

A reservoir-sampling algorithm [26,27] is a simple, unbiased random sampling algorithm for drawing a sample of size n without replacement from a population of size N ($N \geq n$). Vitter [26] has developed a one-pass reservoir-sampling algorithm when the population size (N) is unknown and cannot be determined efficiently. The term “reservoir” defines a storage area j ($j \geq n$, but mostly $j = n$) to store the potential candidates of the sample. The j reservoirs is initialized to store the first j records of the file, that is, all areas of the reservoir pool are initially filled up. Then the algorithm starts scanning the remaining part of the file with a randomly skipping step. The random picked record is evaluated whether to replace the existing one in the reservoir pool. If it passes the test, the position in the reservoir is also randomly selected. The process stops when the end of file has been reached and the records in the reservoir form a simple random sample of the population. The general procedure of reservoir-sampling algorithm [27,28] is given in Figure 1.

The time complexity of the algorithm is shown [26,27] to be O(n (1+ log(N/n)). In the reservoir-sampling algorithm, each record of the file is assigned a uniform (0,1) random number. When the reservoir is needed to be updated, each record in the reservoir has the same chance to be replaced by the new record.

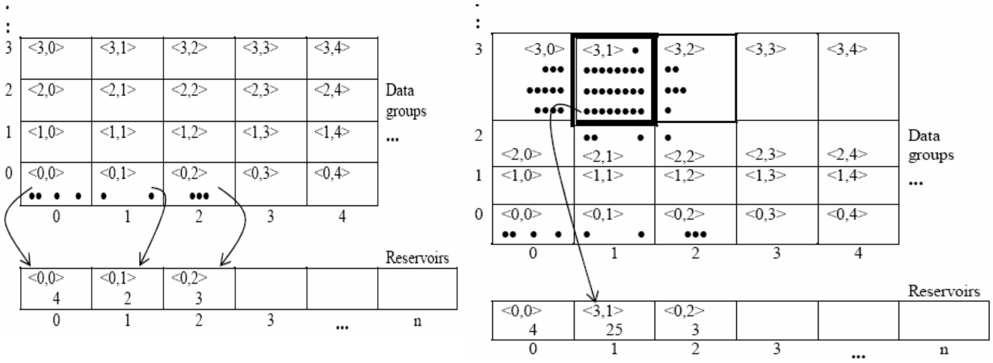
```
Algorithm Reservoir sampling
  Input:   a sequential file of $N$ population
  Output:  a random sample of size $n$ ($n \le N$)
Initialize the reservoir $X_1, ..., X_n$ to be the first $n$ records of the file
Initialize $W$ to be the largest value in a sample of size $n$ from the uniform
  distribution on the interval (0, 1)
While not eof do
    Generate the random variate $S$ to denote the number of records to be skipped
      over before a new record can enter the reservoir
    If (not eof) Then Search for the next potential record to be in the reservoir
                 Else return $X_1, ..., X_n$
    Update $X$ and $W$
```

Fig. 1. Reservoir-sampling algorithm

Our sampling algorithm generalizes the reservoir scheme for the case of data with different density distribution. In our proposed method, the initial step of partitioning data into groups resembles that of Palmer and Faloutsos [21]. But our subsequent steps are not based on hashing scheme in order to avoid the effect of noise and collision problems.

After the initial step of dividing the data space into bins of equal size, the information of the first *n* groups are put into the *n* reservoirs residing in main memory (see Figure 2a). The collected information includes the number of points in each group and the id of the group.

The algorithm performs a single scan on a data set in a random manner controlled by a random variate *S* with the distribution *W*. The density biasing (step 7 in Figure 3) is achieved through the consideration of two consecutive data groups. If the density difference of the two data groups is above some threshold δ (i.e., detecting cluster edge) or the sum of density on both groups is above the threshold value ε (i.e., avoiding noisy cases), then the denser group is a candidate to be included in a sample. This new candidate is put into a reservoir pool at a random position (the reservoir update is pictorially shown in Figure 2b). The density-biased sampling proceeds until the skipping variate *S* reaches the end of the data groups.



(a) initialize the reservoir (b) update reservoir randomly

Fig. 2. Density biasing in a reservoir scheme

```
Algorithm Density-biased reservoir sampling
    Input:   a data set of $N$ objects
    Output:  a density-biased sample of size $n$ ($n \le N$) associated with weight $w$
Partition data into $g$ groups (with group-id $1,2,..., g$), $g \ge n$
Initialize the reservoir $X_1, ..., X_n$ to be the first $n$ <group-id, density>-pairs of the
  data groups
Set $W \leftarrow$ exp(log (random()) / $n$)        // initialize $W$ that will be used in the
                                                   // generation step of random variate $S$
Set $S \leftarrow \lfloor$ log (random()) / log(1-$W$) $\rfloor$
While $S < g$ do
    Read data groups $g_S$ and $g_{S+1}$            // read two consecutive data groups
    If (||density($g_S$) – density($g_{S+1}$)|| > $\delta$ ) OR ((density($g_S$) + density ($g_{S+1}$)) > $\varepsilon$)
                                  // $\delta$ and $\varepsilon$ are predefined density threshold values
    Then $X_{1+\lfloor n * \text{random}()\rfloor} \leftarrow$ <group-id, density> of maximum density{$g_S$ , $g_{S+1}$}
                                  // randomized the reservoir area to be updated
    $W \leftarrow W *$ exp(log (random()) / $n$)      // update $W$ for the skipping process
    $S \leftarrow \lfloor$ log (random()) / log(1-$W$) $\rfloor$    // generate $S$ to denote the number of
                                                   // groups to be skipped over
Return $X_1, ..., X_n$
```

Fig. 3. Density-biased reservoir sampling algorithm

3 Weighted K-Means Algorithm

The classical k-means algorithm [1] is a fast method to perform clustering. The algorithm consists of a simple re-estimation procedure as outlined in Figure 4.

```
Algorithm K-means
    Input:   a set of $n$ data points, and the number of clusters ($K$)
    Output:  centroids of the $K$ clusters
Initialize the $K$ cluster centers
Repeat
        Assign each data point to its nearest cluster center
        Recompute the cluster centers using the current cluster memberships
Until there is no further change in the assignment of the data points to new
  cluster centers
```

Fig. 4. K-means algorithm

The original n data points to be clustered are contained in the dataset $X = \{x_1, ..., x_n\}$. The k-means algorithm partitions n data points into K sets. The assignment of a data point x_i to its nearest cluster center c_j (step 2) is decided on the basis of the membership function, $m(c_j|x_i)$. The function returns either one of the {0,1} values: $m(c_j|x_i) = 1$ if $j = argmin_k||x_i - c_k||^2$; it is zero, otherwise. In step 3, the new centroids of clusters can be computed from all data points x_i in the cluster. The objective function J of the algorithm is to minimize the sum of error squared, $J = \Sigma_{i = 1:n}\, min_{j \in \{1..k\}}\, || x_i - c_j ||^2$.

In k-means algorithm, every data point has equal importance in locating the centroid of the cluster. This property does no longer hold in the case of density-biased sample clustering, for which each data point represents varied density in the original data. Therefore, the clustering algorithm has to consider a weight associated with each data point in the computation of cluster centers. The proposed extension to the k-means algorithm is called weighted k-means. Figure 5 outlines the algorithm.

```
Algorithm Weighted k-means
    Input:   a set of $n$ data points obtained from the density-biased reservoir sampling,
             and the number of clusters ($K$)
    Output:  centroids of the $K$ clusters
1)  Initialize the $K$ cluster centers
2)  Repeat
            Assign each data point to its nearest cluster center according to the
              membership function,
                $m(c_j|x_i) = \frac{|| x_i - c_j ||^{-p-2}}{\Sigma_{j = 1:k} || x_i - c_j ||^{-p-2}}$
3)          For each center $c_j$, recompute the cluster center $c_j$ using the current cluster
              memberships and weights,
                $c_j = \frac{\Sigma_{i = 1:n}\, m(c_j|x_i)\, w( x_i)\, x_i}{\Sigma_{i = 1:n}\, m(c_j|x_i)\, w( x_i)}$
             where $w(x_i)$ is a weight associated with each data point
4)  Until there is no reassignment of data points to new cluster centers
```

Fig. 5. Weighted k-means algorithm

The membership function in the weighted k-means algorithm resembles that of the k-harmonic means algorithm [5]. Zhang [5] also introduces the weight function, $w(x_i)$, in his algorithm to accelerate the recomputation of the new centroids in the next iteration. The weight function in our algorithm, however, is introduced for the different purpose. It represents the density of the original data points.

4 Experiments and Results

We perform two sets of experiments to test the quality of our sampling method, which is the step prior to clustering, and to measure the quality of the weighted k-means algorithm.

4.1 Performance of Density-Biased Reservoir Sampling

We evaluate the performance of the proposed reservoir-based density bias sampling method against the hash-based sampling method [21]. The efficiency regarding memory usage of our reservoir-based sampling method is obviously better than the hash-based method. In the hashing scheme, some amount of memory is needed to store the hashing table in addition to the memory required for storing the drawn sample. Thus, it requires twice the amount of memory comparative to those required by our method.

Effectiveness of the proposed sampling method is examined by measuring the quality of a sample with respect to the number of correctly found clusters. We run clustering using the k-means algorithm. We use a synthetic data generator to generate d-dimensional data sets having *k* clusters and *N* data points. We vary d from 2 to 5, *k* from 2 to 10, and *N* from 5,000 to 100,000.

The measurement *Number of Clusters found* (NC) is the metric defined in [21]. NC is calculated by comparing the distances of the cluster centers found by the clustering algorithm with the true cluster centers. We say that the cluster is found if the calculated distance is less than a predefined threshold (e.g., 0.001).

The results in Figure 6 show the NC when run clustering on various sample sizes with the presence of noise. The reported results are observed from the experiments using 3-dimensional data set having 7 clusters. One cluster contains 50,000 points and the other six clusters contain 500 points. The results obtained from other experiments on data sets with different dimensions, various number of clusters, and varied number of data points are conformed with the one presented in Figure 6, so we omit them for brevity. The experimental results reveal the efficiency of the biased reservoir method especially in the presence of noise.

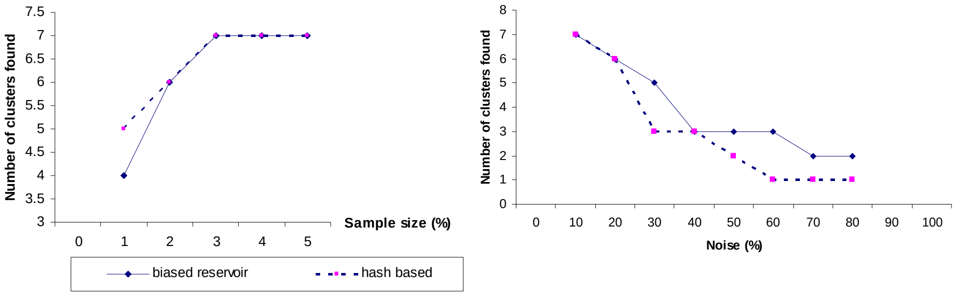


Fig. 6. Finding clusters of 3-dimensional data on various sample sizes, in the presence of noise

4.2 Performance of Weighted K-Means Algorithm

We evaluate the quality of the weighted k-means algorithm against the k-means algorithm by using the squared objective function. Lower value of a squared objective function reflects a better quality on clustering. The experiments perform on the syntactic data sets explained in Section 4.1. The initialization step randomly selects data points as initial cluster centroids. We also consider running time of both algorithms.

The performance evaluation as shown on top of Figure 7 is obtained from running k-means and weighted k-means algorithms on 3-dimensional data sets of sizes varied from 5000, 10000, 20000, 35000, 55000, 75000, to 100000 data points. The number of clusters is set to be 10. The experiments are performed on the PC computer with CPU speed 800 MHz, memory 512 MB. Since all data points are used in weighted k-means algorithm, the weight function is set to be 1. The parameter p in the membership function is set to be 1.3.

The comparison on clustering quality and running time shown at the bottom of Figure 7 reveals the efficiency of running weighted k-means on density-biased sample. The experiments are performed on 10% sample of data with two methods of sampling: simple random sampling (RS) and density-biased reservoir sampling (DBS). The weight function of the weighted k-means algorithm is varied according to the density of the original data.

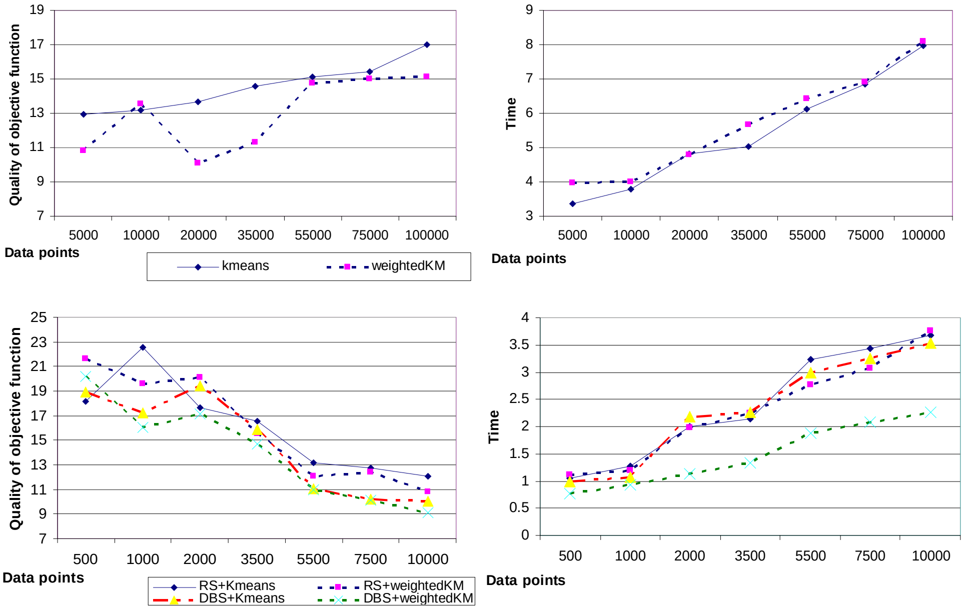


Fig. 7. Performance comparison of weighted k-means against k-means (left) and the running time comparison (right), results on top are experiments running on the whole data set while results at the bottom obtained from running on the sample data

5 Conclusions

The k-means is the simplest and most commonly used clustering algorithm. The simplicity is due to the use of squared error as the stopping criteria, which tends to work well with isolated and compact clusters. Its time complexity depends on the number of data points to be clustered and the number of iteration. We propose a variation of the k-means to better work with a large data set having much difference in cluster density. Our intuition idea is that to cope with massive data set, sampling should be the efficient data reduction method. Since the original data is assumed to be much varied in cluster sizes, density-biased sampling is an appropriate method to preserve the density.

We propose a density biased sampling technique based on the reservoir method. The inherent advantage of efficient memory usage in the reservoir scheme is adopted and extended with the additional capability of dealing with data that are much different in density distribution. The proposed technique is designed to lessen the effect of noise as it is the case in the hash-based approach. The experimental results have shown that the proposed method is as good as the hash-based method in discovering correct number of clusters. Our method, moreover, is less sensitive to noisy data even when the percentage of noise is greater than 20.

We also develop the weighted k-means algorithm to better cluster a sample data biased by its density. The results demonstrate the efficiency of the algorithm. The evaluation of the proposed methods on real large databases and the consideration of outliers are our future work.

Acknowledgements

This research has been supported by grants from the Thailand Research Fund (TRF, MRG4780170), and the National Research Council. The Data Engineering and Knowledge Discovery Research Unit is fully supported by the research grants from Suranaree University of Technology.

References

1. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, Vol.1. University of California Press (1967) 281-297
2. Hamerly, G., Elkan, C.: Alternatives to the k-means algorithm that find better clusterings. In Proceedings of 11th ACM CIKM International Conference on Information and Knowledge Management (2002) 600-607
3. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York (1981)
4. Pellog, D., Moore, A.: Accelerating exact k-means algorithms with geometric reasoning. In Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (1999) 277-281
5. Zhang, B.: Generalized k-harmonic means - boosting in unsupervised learning. Technical Report HPL-2000-137. Hewlett-Packard Labs (2000)
6. Bradley, P.S., Fayyad, U.M.: Refining initial points for k-means clustering. In Proceedings of the 15th International Conference on Machine Learning (1998) 91-99

7. Meila, M., Heckerman, D.: An experimental comparison of model-based clustering methods. Machine Learning 42(2001) 9-29

8. Pena, J., Lozano, J., Larranaga, P.: An empirical comparison of four initialization methods for the k-means algorithm. Pattern Recognition Letters 20(1999) 1027-1040

9. Likas, A., Vlassis, N., Verbeek, J.: The global k-means clustering algorithm. Technical Report IAS-UVA-01-02. Computer Science Institute, University of Amsterdam, The Netherlands (2001)

10. Pelleg, D., Moore, A.: X-means: Extending k-means with efficient estimation of the number of clusters. In Proceedings of the 17th International Conference on Machine Learning (2000) 727-734

11. Sand, P., Moore, A.: Repairing faulty mixture models using density estimation. In Proceedings of the 18th International Conference on Machine Learning (2001)

12. Sander, J., Ester, M., Kriegel, H.-P., Xu, X.: Density-based clustering in spatial databases: The algorithm GDBSCAN and its application. Data Mining and Knowledge Discovery 2(1998) 169-194

13. Guha, S., Rastogi, R., Shim, K.: CURE: An efficient clustering algorithm for large databases. In Proceedings of the ACM SIGMOD International Conference on Management of Data (1998) 73-84

14. Ng, R.T., Han, J.: Efficient and effective clustering methods for spatial data mining. In Proceedings of International Conference on Very Large Data Bases (1994) 144-155

15. Zhou, S., Zhou, A., Cao, J., Wen, J., Fan, Y., Hu., Y: Combining sampling technique with DBSCAN algorithm for clustering large spatial databases. In Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining (2000) 169-172

16. Nanopoulos, A., Theodoridis, Y., Manolopoulos, Y.: C^2P: Clustering based on closest pairs. In Proceedings of International Conference on Very Large Data Bases (2001) 331-340

17. Singh, G., Rajagopalan, S., Lindsay, B.: Random sampling techniques for space efficient of large data sets. In Proceedings of ACM SIGMOD International Conference on Management of Data (1999)

18. Toivonen, H.: Sampling large databases for association rules. In Proceedings of International Conference on Very Large Data Bases (1996) 134-145

19. Thompson, S.K., Seber, G.A.F.: Adaptive Sampling. John Wiley & Sons, New York (1996)

20. Olken, F., Rotem, D.: Sampling from spatial databases. In Proceedings of International Conference on Data Engineering (1993) 199-208

21. Palmer, C., Faloutsos, C.: Density biased sampling: An improved method for data mining and clustering. In Proceedings of ACM SIGMOD International Conference on Management of Data (2000) 82-92

22. Nanoupoulos, A., Theodoridis, Y., Manolopoulos, Y.: An efficient and effective algorithm for density biased sampling. In Proceedings of 11th International Conference on Information and Knowledge Management (2002) 63-68

23. Kollios, G., Gunopulos, D., Koudas, N., Berchtold, S.: Efficient biased sampling for approximate clustering and outlier detection in large data sets. IEEE Transactions on Knowledge and Data Engineering 15(2003) 1-18

24. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: An efficient data clustering method for very large databases. In Proceedings of ACM SIGMOD International Conference on Management of Data (1996) 103-114

25. Zipf, G.K.: Human Behavior and Principle of Least Effort: An Introduction to Human Ecology. Addison Wesley, Cambridge, MA (1949)

26. Vitter, J.S.: Random sampling with a reservoir. ACM Transactions on Mathematical Software 11(1985) 37-57

27. Li, K.-H.: Reservoir-sampling algorithms of time complexity O(n(1 + log(N/n))). ACM Transactions on Mathematical Software 20(1994) 481-493

28. Devroye, L.: Non-Uniform Random Variate Generation. Springer-Verlag, New York (1986)