Performance Analysis of Clustering Algorithms for Information Retrieval in Image Databases

Tak Kan Lau and Irwin King
Department of Computer Science and Engineering, The Chinese University of Hong Kong
Shatin, New Territories, Hong Kong, {tklau, king}@cse.cuhk.edu.hk

Abstract

In image databases, a good indexing method makes nearest-neighbor retrieval of images accurate and efficient. Since existing alphanumeric indexing methods are not particularly suitable in image databases, researchers have proposed new methods for indexing by clustering methods. In this paper, we analyze the performance of two unsupervised neural network clustering algorithms *Competitive Learning (CL)* and *Rival Penalized Competitive Learning (RPCL)* together with *k-means* and *VP-tree* for image database indexing. We present some performance experiments to measure their accuracy and efficiency. Based on the experimental results, we concluded that RPCL and CL are good information retrieval in image database.

1. Introduction

In this information age, good information management tools are required for handling information efficiently and effectively. Image databases are the tools used for digital image information management. They often make use of the indexing methods to speed up the information retrievals and produce accurate results.

A good indexing method is a key component in an image database for efficient and accurate information retrieval. Nowadays, alphanumeric data indexing techniques are already well developed. However, image databases use mainly image contents such as color, texture, and shape for retrieval. Those indexing methods are not particularly suitable for this use. So, people have begun to develop some new indexing methods for content-based retrieval in image databases.

There are several indexing methods designed for image database indexing such as R-tree [1], Quadtree [7], and VP-tree [9]. For example, R-tree is a generalization of B-tree for multidimensional database objects. VP-tree cuts the database object space into subspaces according to the distances between the database objects and vantage points for indexing.

In an image database, we often perform nearest-neighbor search to retrieve images with similar contents to the query. A nearest-neighbor search in an image database is a traversal algorithm of an indexing structure to find the node which contains a set of feature vectors closest to the query under a distance function. The indexing methods mentioned in the last paragraph work fine for many cases, but all of them seem to fail to retrieval similar database objects when a nearest-neighbor query lies on the boundary of the partition. By these methods, some of the database objects may be partitioned wrongly into another partition for a boundary nearest-neighbor query. Therefore, the result of the query may not be so accurate.

Some researchers proposed to use clustering methods for image database indexing. The authors [3, 2] proposed to use *Competitive Learning* and *k-means* whereas one of the authors (King) and Xu [4] proposed to use *Rival Penalized Competitive Learning* to generate clusters for indexing. Since the set of feature vectors often forms a sparse multidimensional feature space, it is natural to assume that there exists an underlying distribution of these vectors. Usually, the distribution is not uniform. As a result, we may group feature vectors together that are generally retrieved as a whole in response to a request query. Clustering is a mutually exclusive partitioning process of the feature vector space in a meaningful way for the application domain context. With the clusters, we may perform nearest-neighbor search efficiently. These clustering methods give better results for the boundary queries than R-tree and VP-tree because they pay attention to the input distribution of the database objects to product natural clusters, but the R-tree and VP-tree indexing methods do not.

In this paper, we analyze the performance of two unsupervised neural network clustering methods *Competitive Learning (CL)* [6] and *Rival Penalized Competitive Learning (RPCL)* [8] which is a variant of competitive learning, a statistical clustering technique *k-means* [5], and a traditional indexing method VP-tree for image database indexing. We use the Recall and Precision values to measure the accuracy of the meth-

0-7803-4859-1/98 $10.00©1998 IEEE

ods for information retrieval and use the time spent in the pre-processing parts including clustering and indexing of the feature vectors to evaluate the efficiency of the methods.

For the rest of the paper, we will briefly describe how to build indexing structures from the clusters generated by the clustering methods in Section 2. Section 3 will present some experiments on these methods for image database indexing and a brief discussion on their performance. A conclusion will be given in Section 4.

2. Using clustering techniques for image database indexing

In this section, we describe how to make use of the clusters generated by the clustering methods k-means, CL, and RPCL to build indexing structures for image database indexing. After clustering by these methods, we may build indexing structures based on the clusters.

There are two approaches: *Hierarchical Approach* and *Hierarchical Approach* to perform top-down clustering and build indexing structures.

The hierarchical approach transforms a feature vector space into a sequence of nested partitions. It partitions the vectors which are in a partition of the previous level (see Figure 1). Quadtree and VP-tree use the hierarchical approach.

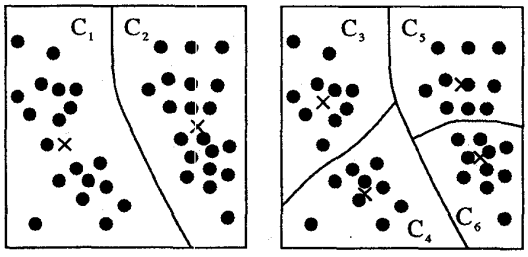

Figure 1. Hierarchical clustering. C_3 and C_4 are the clusters inside C_1. C_5 and C_6 are the clusters inside C_2. The dots represent the database objects. The crosses represent the centers.

The hierarchical approach can be formulated as follows. Let the feature vector set X with n vectors be $X = \{x_1, x_2, \ldots, x_n\}$. A cluster, C, of X breaks X into subsets $C_1, C_2, \ldots, C_m$ satisfying the following: $C_i \cap C_j = \emptyset, i \neq j$ and $C_1 \cup C_2 \cup \ldots \cup C_m = X$. Cluster B is nested into cluster C if every B's component is a proper subset of a C's component. A hierarchical clustering is a sequence of clusters in which each cluster is nested into the previous cluster in the sequence.

After the clustering, there exists a mapping function that maps the generated clusters to a binary indexing structure. For example, all the feature vectors are in one cluster at the root level and there are 2^i subtrees (clusters) at depth i (see Figure 2).

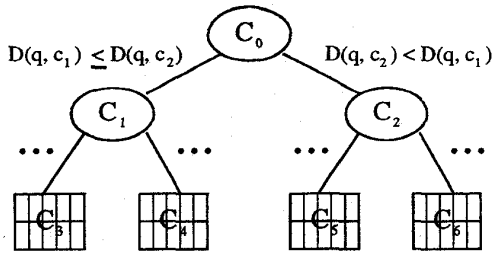

Figure 2. The indexing structure for the hierarchical clustering in Figure 1. C_0 is the root node which contains all the database objects. $D(q, c_i)$ means the distance between nearest-neighbor query q and the center c_i of cluster C_i.

At the top level, a nearest neighbor query q is compared to the centers of the clusters in the immediate lower level. The cluster with center closest to the query point q is selected. The elements in the selected cluster will be the result of the query if they satisfy the criteria of the nearest neighbor search. Otherwise, the search will proceed to the lower levels.

The non-hierarchical approach considers the whole feature vector space each time to cluster. Based on the generated clusters, we may make use of the inverted file structure to do indexing for the feature vectors (see Figures 3 and 4). Given the feature vector space having only two clusters C_1 and C_2 with centers c_1 and c_2 respectively, for example, a feature vector v will belong to C_1 if $D(v, c_1) < D(v, c_2)$ and it will be indexed as "C_1". D is a distance function and is defined as: $D(x, y) = ||x - y|| = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}$. According to the indexing structure, a nearest-neighbor query q is compared to all the cluster centers. All the vectors belonged to the cluster whose center is closed to q will be retrieved. In our work, the k-means, CL, and RPCL methods use the non-hierarchical approach.

3. Performance experiments

We used some experiments to evaluate the performance of the clustering methods: k-means, CL, RPCL, as well as an indexing method VP-tree for information retrieval in image databases based on their accuracy and efficiency. We conducted three sets of experiments on three different databases which are consisted of:

1. synthetic data in gaussian distribution
2. synthetic data in uniform distribution

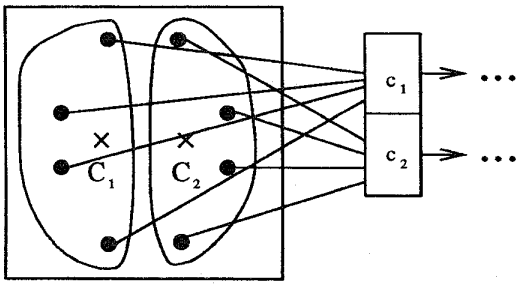

Figure 3. Two clusters generated by non-hierarchical clustering. The dots are the database objects whereas the crosses are the cluster centers. An inverted file structure is used for indexing.

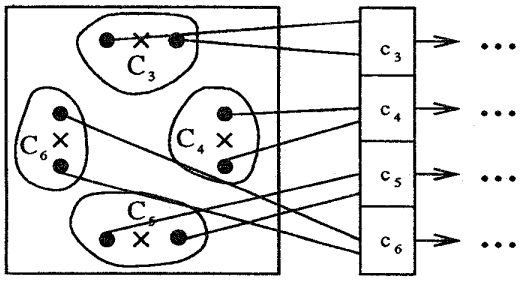

Figure 4. Four clusters generated by non-hierarchical clustering.

3. feature vectors extracted from synthetic images

respectively. All of the experiments were conducted on an Ultra Sparc 1 machine running Matlab V4.2c.

We used two performance measurements : *Recall* and *Precision* in the experiments to measure the accuracy of the methods. They are defined as:

$$\text{Recall} = \frac{\text{Number of target images retrieved}}{\text{Number of target images}} \quad (1)$$

$$\text{Precision} = \frac{\text{Number of target images retrieved}}{\text{Number of images retrieved}} \quad (2)$$

For the gaussian distributed database objects, the equations can be rewritten as below. Given the set of *a priori* clusters, $C = \{c\}_1^n$ and the set of generated clusters $C' = \{c'\}_1^m$, the performance measurements Recall and Precision are defined as :

$$\text{Recall} = \sum_{c_i \in C \wedge c'_j \in C'} \frac{c_i \cap c'_j}{\#c_i} \quad (3)$$

$$\text{Precision} = \sum_{c_i \in C \wedge c'_j \in C'} \frac{c_i \cap c'_j}{\#c'_j} \quad (4)$$

where $\#c_i$ denotes the number of elements in c_i.

3.1. Experiment 1

In the first experiment, we tested the methods in producing indexing structures using 1024 2-dimensional synthetic feature vectors which were in gaussian distribution. Let $\mu = (\mu_1, \mu_2, \ldots, \mu_n)$ and $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$, we generated the input distribution of the feature vectors from the mixture of n Gaussian distributions $N(\mu, \sigma^2)$ with the generating function defined as $g(x) = 1/(\sigma\sqrt{2\pi}) \exp[-[(x-\mu)^2/2\sigma^2]]$, $-\infty < x < \infty$. In our experiments, we simply used a constant 0.1 for σ and let $n = 2, 4, 8, 16$, and 32. For each input distribution, different number of clusters are generated for the input feature vectors.

Based on the same set of feature vectors, we conducted 20 trails with different initial starting points of the centers of the to-be generated clusters for all the methods. We then calculated the average performance of the four methods using equations 3 and 4. Tables 1 and 2 show the results for the average Recall and Precision measurements.

Moreover, we also kept the time used for the pre-processing which includes clustering and indexing of the feature vectors. Since this time is independent to the input distribution of the feature vector space, we show only the time used for the pre-processing of the 1024 feature vectors with 8 Gaussian mixtures of the input distribution in Figure 5.

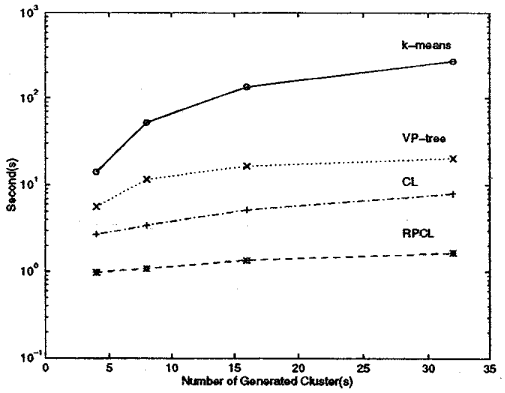

Figure 5. Time used for the pre-processing of the 1024 feature vectors with 8 Gaussian mixtures of the input distribution.

3.2. Experiment 2

In the second experiment, we tested the methods in producing indexing structures using 1024 2-

Mixture Groups	Generated Clusters (k-means, CL, RPCL, VP-tree) 4	8	16	32
2	.64, .57, .51, .50	.39, .28, .27, .25	.20, .17, .16, .13	.16, .10, .14, .06
4	.81, .86, .79, .81	.53, .48, .44, .41	.45, .24, .24, .22	.36, .13, .21, .11
8	.78, .71, .72, .68	.51, .51, .53, .54	.37, .37, .29, .25	.32, .21, .20, .13
16	.82, .85, .85, .86	.65, .63, .65, .61	.45, .43, .39, .36	.28, .28, .26, .18
32	.78, .78, .78, .73	.60, .60, .59, .55	.44, .43, .44, .38	.30, .32, .29, .26

Table 1. Recall table for experiment 1.

Mixture Groups	Generated Clusters (k-means, CL, RPCL, VP-tree) 4	8	16	32
2	.88, .89, .94, .99	.90, .95, .95, .99	.96, .98, .98, .99	.98, .98, .99, .99
4	.73, .66, .65, .58	.73, .70, .70, .67	.81, .73, .73, .71	.88, .77, .81, .76
8	.47, .26, .34, .22	.47, .43, .46, .35	.44, .44, .51, .43	.44, .49, .58, .48
16	.13, .13, .13, .13	.19, .17, .18, .17	.23, .21, .22, .20	.29, .26, .29, .25
32	.07, .06, .07, .06	.09, .09, .09, .09	.14, .12, .13, .12	.19, .17, .19, .16

Table 2. Precision table for experiment 1.

dimensional synthetic feature vectors which were in uniform distribution. Based on the same set of feature vectors, we conducted 20 trails with different initial starting points of the centers of the to-be generated clusters for all the methods. We then calculated the average performance of the four methods using equations 1 and 2. Figures 6 and 7 show the results for the average Recall and Precision measurements with different numbers of generated clusters. Figure 8 shows the pre-processing time of the input feature vectors.

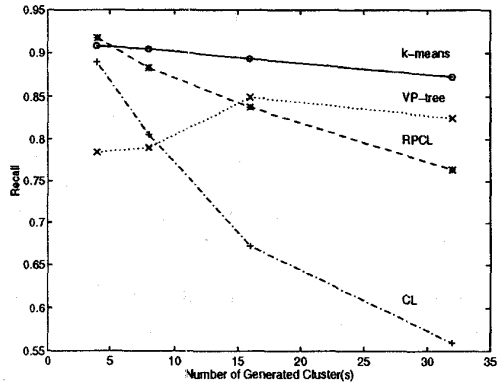

Figure 6. Recall results for experiment 2.

3.3. Experiment 3

In the third experiment, we used the feature vectors extracted from 400 synthetic images to test the performance of the methods in producing indexing structures. First, we generated the 400 synthetic images.

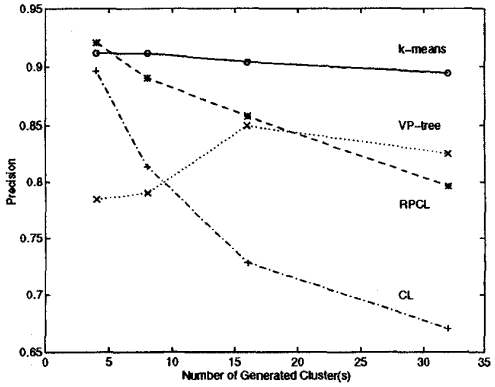

Figure 7. Precision results for experiment 2.

Each of them is consisted of two randomly selected colors in different percentages and with the size of 128 × 128 pixels. It took about 105 seconds for the image generation process. Then, for each image, we quantized the color space evenly into 27 colors and spent about 0.016 seconds to extract an 27-bucket color histogram from it. All the 400 histograms formed a 27-dimensional feature vector space.

Based on the 400 extracted feature vectors, we conducted 20 trails with different initial starting points of the centers of the to-be generated clusters for all the methods. We then calculated the average performance of the four methods using equations 1 and 2. Figures 9 and 10 show the results for the average Recall and Precision measurements with different numbers of gen-

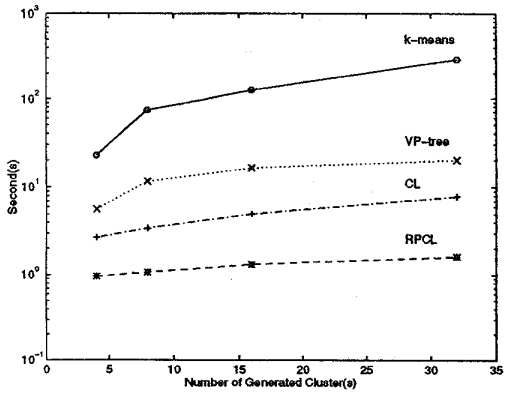

Figure 8. Time used for the pre-processing parts for the four methods in experiment 2.

erated clusters. Moreover, Figure 11 shows the time used for the pre-processing of the input feature vectors.

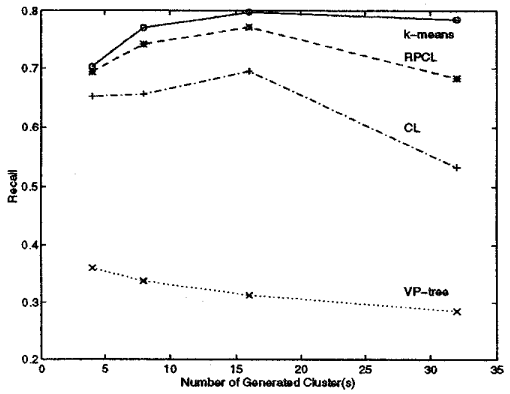

Figure 9. Recall results for experiment 3.

3.4. Discussion

Based on the results of the above experiments, a discussion of the performance of the 4 methods for information retrieval in image databases is given below:

- **Boundary problem for hierarchical and non-hierarchical clustering.** For experiment 1, when the number of Gaussian mixtures of the input distribution is greater than or equal to the number of generated clusters, the Recall values of the VP-tree method are often lower than the others. It is due to VP-tree cannot handle the boundary problem well. This is a problem when a requested nearest neighbor query falls near the partition boundary. Since VP-tree does not pay attention to the input distribution, a similar feature vector near the query may be clustered in another partition. Therefore, the Recall values of the VP-tree method are decreased.

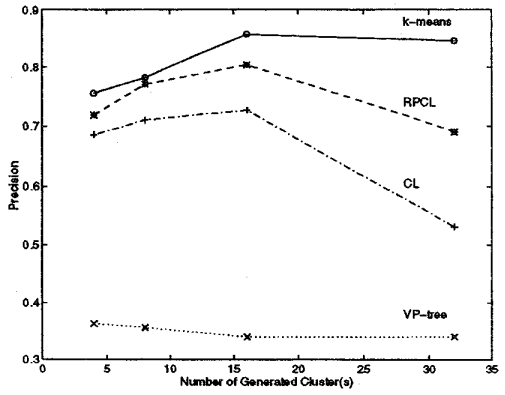

Figure 10. Precision results for experiment 3.

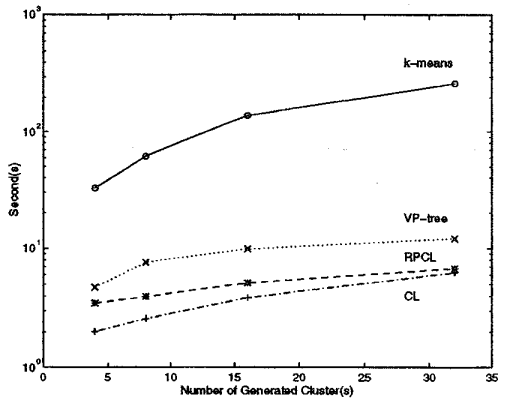

Figure 11. Time used for the pre-processing parts for the four methods in experiment 3.

- **Problem in indexing gaussian distributed data.** In experiment 1, there is a problem for the k-means, CL, and RPCL methods when the number of generated clusters is greater than the number of the Gaussian mixtures. We find that the Recall values are relatively low in this case. It is because multiple generated clusters can be bunched together spatially. This leads to an incorrect assessment of clusters since only a few target images can be retrieved.

- **High Recall and Precision for uniform distributed data.** In experiment 2, we find that

the overall Recall and Precision values of the four methods are higher than those in experiment 1. It is because there is no explicit clusters found in the distribution of the input feature vectors. The effect of boundary problem is not high in this case. Moreover, all the four methods seem to partition the feature space evenly and no two cluster centers are bunched together. So, they give higher Recall and Precision results than those in experiment 1.

- **K-means: an accurate but slow method.** From experiments 1 and 2, k-means gives the best average Recall and Precision performance among the four methods but it is the slowest. It is because the k-means algorithm often recomputes the cluster centers in the partitioning process. Hence, it is computationally intensive, but more accurate.

- **RPCL: a fast method with satisfactory Recall and Precision.** From the first two experiments, RPCL and CL give satisfactory results and they are faster than k-means since they use the competitive learning technique to increase the speed of partitioning. Moreover, we find that RPCL is overall faster than CL because RPCL is an improved version of CL.

- **Result for the extracted feature vectors.** Experiment 3 was conducted based on some extracted feature vectors. The experimental result shows the same properties mentioned above in this discussion section. Since the distributions of the feature vectors in the first two experiments represent two extreme cases: gaussian and uniform, the distribution of the extracted feature vectors must be in between these two cases. Therefore, we find that the overall Recall and Precision values are higher than the gaussian one but lower that the uniform one. Moreover, k-means gives the best results but slow. RPCL and CL give satisfactory results but much faster than k-means. Because of the boundary problem, VP-tree give the worst results among the the four methods.

4. Conclusion

We have analyzed the performance of two unsupervised neural network clustering methods CL and RPCL together with k-means, and a traditional indexing method VP-tree for information retrieval in image databases. Form the experiments, we find that the clustering indexing methods give better overall performance than VP-tree because the effect of boundary problem for the clustering methods is lower. Moreover, the k-means method has the best average performance but it is slow. As a result, it is concluded that RPCL followed by CL are the best methods for image database indexing among the four analyzed methods.

5. Acknowledgement

The work was supported in part by the Hong Kong Research Grant Council # CUHK 4176/97E and IDTC Grant # AF/17/95.

References

[1] A. Guttman. "R-trees: A Dynamic Index Structure for Spatial Searching". *ACM SIGMOD*, 14(2):47–57, 1984.

[2] I. King and T. K. Lau. "Comparison of Several Partitioning Methods for Information Retrieval in Image Databases". In *Proceedings of the 1997 International Symposium on Multimedia Information Processing (ISMIP'97)*, pages 215–220, 1997.

[3] I. King and T. K. Lau. "Competitive Learning Clustering for Information Retrieval in Image Databases". In *Proceedings of the 1997 Internation Conference on Neural Information Processing (ICONIP'97)*, pages 906–909, 1997.

[4] I. King, L. Xu, and L. W. Chan. "Using Rival Penalized Competitive Clustering for Feature Indexing in Hong Kong's Textile and Fashion Image Database". [Submitted to IJCNN'98].

[5] J. B. MacQueen. "Some Methods for Classification and Analysis of Multivariate Observations". In *Proc. Symp. Math. Statist. and Probability, 5th, Berkeley*, pages 281–297, 1967.

[6] D.E. Rumelhart and D. Zipser. "Feature discovery by competitive learning". *Cognitive Science*, 9:75–112, 1985.

[7] H. Samet. "The quadtree and related hierarchical data structures". *Computer Surveys*, 16(2), 1984.

[8] L. Xu, A. Krzyżak, and E. Oja. "Rival penalized competitive learning for clustering analysis, RBF net, and curve detection". *IEEE Trans. on Neural Networks*, 4(4):636–649, 1993.

[9] P. N. Yianilos. "Data structures and algorithms for nearest neighbor search in general metric spaces". In *Proc. of the 4th Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 311–321, 1993.