Comparison of Several Partitioning Methods 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
Tel: (852) 2609-8480
Fax: (852) 2603-5024

Abstract

Efficient and accurate information retrieval is one of the main issues in image databases. Since traditional alphanumeric indexing methods are not particularly suitable for image database indexing, new indexing methods are designed specially for image retrieval. In this paper, we are going to discuss four partitioning methods: *VP-tree*, *k-means*, *Competitive Learning* (CL), and *Rival Penalized Competitive Learning* (RPCL) for image database indexing. Our aim is to evaluate and compare the performances of these methods for information retrieval in image database based on the performance measurements: Recall, Precision, and Speed. From the result of some performance experiments, it is concluded that RPCL followed by CL are the most appropriate to be used for image retrieval among these four methods.

1 Introduction

Efficient and accurate information retrieval is a key issue in image databases. Since image databases use image contents such as color, texture, sketch, and shape for retrieval, the traditional alphanumeric indexing methods are not particularly suitable. So, new indexing methods are developed for fast and accurate information retrieval in image databases. In this paper, we will discuss some partitioning methods for image database indexing.

In an image database, we often perform nearest neighbor search to retrieve images with similar contents or features 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 vector under a distance function. Since the feature vectors form 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. This leads to clustering of features vectors.

Clustering is a mutually exclusive partitioning process of the feature space of feature vectors in a meaningful way for the application domain context. With the clusters, we may perform nearest neighbor search efficiently.

There are many partitioning methods which generate suitable clustering of the feature vectors. Based on the generated clusters, an indexing structure can be produced for accessing database objects with similar features. In this paper, we are going to discuss the performances of four partitioning methods: *VP-tree* [6], *k-means* [3, 1], *Competitive Learning (CL)* [4, 2], and *Rival Penalized Competitive Learning (RPCL)* [5]. Our aim is to compare and evaluate these methods to be used for information retrieval in image databases based on the *Recall*, *Precision* performances, and their *speeds* for producing clusters for indexing.

In Section 2, we will describe these four partitioning methods briefly and show how to build indexing structures from generated clusters. Then, we will present some performance experiments on these methods for information retrieval in image databases in Section 3. Some discussions on their performances will be given in Section 4.

2 Four Partitioning Methods for Image Database Indexing

We use the four partitioning methods to produce clusters for image database indexing. Here is a brief description of these methods.

- **VP-tree** : The VP-tree method partitions the feature vector space based on the distances between the feature vectors and a selected vantage point. According to the median of these distances, the whole feature space are divided into two sets: near vector set and far vector set. The process will continue in both sets individually.

- **k-means** : The k-means method groups the feature vectors into k clusters. First, take k feature vectors randomly for the centers of k clusters. Then, assign each of the remaining vectors to the cluster with nearest centroid. After each assignment, recompute the center of that gaining cluster. After all the vectors have been assigned, keep all the cluster centers and make one more assignment pass. If any center changes, go to another pass, otherwise, stop.

- **CL** : This is a neural network learning algorithm for partitioning. Initially, there is a set of neurons with some randomly distributed synaptic weights. For each of the feature vectors, the neurons compete among themselves and only one of them will win the competition. The learning rule will then move the synaptic weight vector of the winning neuron toward the feature vector. After all the feature vectors are processed, the synaptic weight vectors of the neurons will be the cluster centers of the feature vectors.

- **RPCL** : RPCL is a variant of CL. Instead of moving only the winning neuron, RPCL moves also the first runner-up neuron away from each of the input feature vector.

After generating all the clusters, we may build an indexing tree based on them. Given a bipartite graph $G = (V, E)$, an indexing tree has the following properties: (1) there is a root node which is the only node at the top of the tree, and (2) the remaining nodes are partitioned into n disjoint sets $G_1, \ldots, G_n$. Each of the trees $G_1, \ldots, G_n$ is a tree itself and is a subtree of the root. When $n = 2$, the indexing tree is a binary indexing tree. Each subtree in the binary tree will be a cluster partition.

There are two approaches to perform top-down partitioning: (1) *hierarchical approach* and (2) *non-hierarchical approach*. The first approach partitions the feature vectors which are in the subset partitioned in the previous level. The second approach considers the whole feature space each time to partition instead. For the above four methods, VP-tree uses the hierarchical approach whereas the other methods use the non-hierarchical approach.

After the partitioning, 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. At the top level, a nearest neighbor query $\hat{x}$ is compared to the centers of the clusters in the immediate lower level. The cluster with center closest to the query point $\hat{x}$ 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.

3 Performance Experiments

We used some experiments to evaluate the performances of the four partitioning methods for information retrieval in image databases. The experiments were conducted on an Ultra Sparc 1 machine running V4.2c. We tested the methods in producing indexing structures using 2560

3-dimensional synthetic feature vectors. 3D vectors are used because some of image features are 3-dimensional, e.g., RGB color. 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.

We used two performance measures : *Recall* and *Precision* in the experiments. Given the set of *a priori* clusters, $C = \{c\}_1^n$ and the set of calculated clusters by the partitioning methods, $C' = \{c'\}_1^m$, the performance measurements Recall and Precision are defined as :

$$\text{Recall} = \sum_{c_i \in C \wedge c'_j \in C'} \frac{c_i \cap c'_j}{\#c_i} = \frac{\text{Number of target images retrieved}}{\text{Number of target images}} \tag{1}$$

$$\text{Precision} = \sum_{c_i \in C \wedge c'_j \in C'} \frac{c_i \cap c'_j}{\#c'_j} = \frac{\text{Number of target images retrieved}}{\text{Number of images retrieved}} \tag{2}$$

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

Mixture	**Generated Clusters (VP-tree, k-means, CL, RPCL)**			
Groups	**4**	**8**	**16**	**32**
2	0.50, 0.67, 0.54, 0.54	0.25, 0.58, 0.26, 0.25	0.13, 0.52, 0.19, 0.13	0.06, 0.51, 0.17, 0.11
4	0.95, 0.96, 0.95, 0.96	0.48, 0.65, 0.48, 0.49	0.24, 0.45, 0.22, 0.25	0.12, 0.20, 0.16, 0.13
8	0.80, 0.96, 0.81, 0.90	0.49, 0.84, 0.80, 0.80	0.29, 0.61, 0.51, 0.41	0.16, 0.34, 0.30, 0.23
16	0.59, 0.78, 0.78, 0.78	0.50, 0.85, 0.71, 0.81	0.36, 0.71, 0.58, 0.61	0.20, 0.37, 0.42, 0.37
32	0.78, 0.77, 0.79, 0.81	0.58, 0.79, 0.65, 0.73	0.44, 0.55, 0.50, 0.58	0.30, 0.45, 0.38, 0.38

Table 1: Recall Table.

Mixture	**Generated Clusters (VP-tree, k-means, CL, RPCL)**			
Groups	**4**	**8**	**16**	**32**
2	1.00, 1.00, 1.00, 1.00	1.00, 1.00, 0.99, 1.00	1.00, 1.00, 1.00, 1.00	1.00, 1.00, 1.00, 1.00
4	0.38, 0.58, 0.57, 0.60	0.56, 0.60, 0.65, 0.63	0.66, 0.59, 0.68, 0.73	0.75, 0.74, 0.74, 0.78
8	0.20, 0.24, 0.23, 0.23	0.26, 0.35, 0.36, 0.38	0.44, 0.44, 0.49, 0.53	0.60, 0.56, 0.55, 0.62
16	0.11, 0.12, 0.12, 0.12	0.16, 0.21, 0.18, 0.20	0.21, 0.30, 0.22, 0.28	0.32, 0.41, 0.27, 0.35
32	0.05, 0.06, 0.06, 0.06	0.08, 0.10, 0.08, 0.09	0.11, 0.13, 0.11, 0.12	0.16, 0.16, 0.15, 0.17

Table 2: Precision Table

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 four methods except VP-tree. We ran 1 trail instead for VP-tree since it always gives the same result for the same set of feature vectors. We then calculated the average performances 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 partitioning. Since this time is independent to the input distribution of the feature vector space, we shows only the time used to generate clusters for the 2560 feature vectors with 8 Gaussian mixtures in Figure 1.

4 Discussion

Table 3 summaries the performances of the four methods. Here is a brief discussion.

- K-means gives the best average Recall and Precision performances among the four methods but it is slowest. It is because k-means often recompute the centers of the clusters in the partitioning process. It will give better partitioning result but it is computationally slow.

- 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, it is obvious that RPCL is faster than CL because RPCL is an improved version of CL.
- When the number of Gaussian mixtures are 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 well for the case 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.
- 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 may 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. However, this problem rarely occurs because the number of natural clusters in the input distribution is quite large in practice.

In conclusion, we have discussed four partitioning methods for image database indexing. Our performance experiments have shown that RPCL gives the best overall performance followed by CL for information retrieval in image databases among these methods. If better Recall and Precision performance is wanted and there is plenty of time, k-means is a good choice.

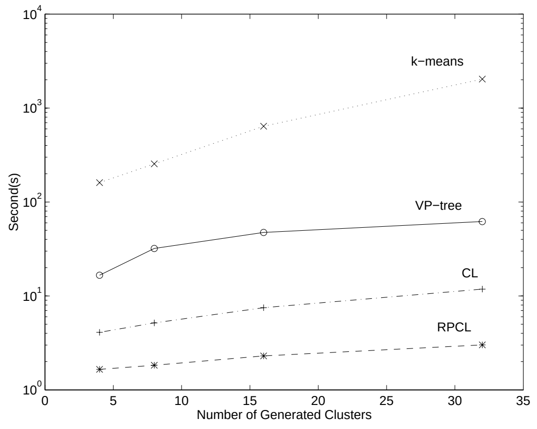

Figure 1: Time used to generate clusters for 2560 feature vectors with 8 Gaussian mixtures.

Measure	VP-tree	k-means	CL	RPCL
Recall	lowest	highest	middle	middle
Precision	lowest	high	high	high
Speed	middle	lowest	high	highest

Table 3: Comparison of the performances of the four methods.

Acknowledgement

The work was supported in part by the Hong Kong Research Grant Council # CUHK 485/95E and IDTC Grant # AF/17/95.

References

[1] M. R. Anderberg. *"Cluster Analysis for Applications"*. Academic Press, New York, 1973.

[2] S. Haykin. *"Neural networks: a comprehensive foundation"*. Macmillan, New York, 1994.

[3] 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.

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

[5] 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.

[6] P. N. Yianilos. "Data structures and algorithms for nearest neighbor search in general metric spaces". In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 311–321, January 1993.