Competitive Learning Clustering for Information Retrieval in Image Databases

Irwin King[1] and Tak Kan Lau[1]

[1]Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong. {king,tklau}@cse.cuhk.edu.hk

December 1, 1998

Abstract

Efficient and accurate information retrieval is a key issue in image databases. Since image databases use image features for retrieval, traditional alphanumeric indexing methods are not particularly suitable for content-based retrieval. Therefore, new indexing methods must be designed and implemented specifically for image retrieval. In this paper, we propose to use competitive learning clustering algorithm to produce an indexing structure for Montage, which is an image database supporting content-based retrieval using *color*, *texture*, *sketch*, and *shape* for Hong Kong's fashion, textile, and clothing industry. Competitive learning is a stochastic and efficient clustering method which provides good cluster center approximation for image database indexing. Using synthetic data, we demonstrate the Recall and Precision performance of nearest neighbor feature retrieval based on the indexing structure generated by competitive learning clustering and show that the algorithm works well.

1 Introduction

Efficient and accurate information retrieval is one of the main issues in image databases. A good indexing structure organizes the data in the database in such a way so as to make the retrieval computationally efficient. But, how can we produce a good indexing structure for image retrieval?

Problem Defined

Let $DB = \{I_i\}_{i=1}^n$ be a set of image objects. With a set of parameters $\theta = \{\theta_1, \theta_2, \ldots, \theta_m\}$, a feature extract function is defined as $f : I \times \theta \rightarrow \mathcal{R}^d$ which extracts a real-valued d-dimensional vector. Moreover, we may let a random variable X to be the feature vector extract form DB and $x_i, i = 1, 2, \ldots, n$ to be the instance of the feature vector.

In an image database, we may often perform similar matching by using a query to retrieve images with similar features in the database. Given a similar matching query $\hat{x}$, the database will retrieve the set $\{x \mid 0 \leq D(x, \hat{x}) \leq \epsilon\}$ where ϵ is a tolerance bound for similarity matching and $D(\cdot, \cdot)$ is a distance function for similarity measurement. In our case, we use the L_2-norm (Euclidean distance) to define the function D as : $D(x, y) = \|x - y\| = \sqrt{\sum_{i=1}^d (x_i - y_i)^2}$.

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 that satisfy the criteria mentioned in the last paragraph. 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 together in response to a request query. This leads to clustering of feature 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.

The problem is then to find an efficient method to generate a suitable clustering of the feature vectors so that, based on the generated clusters, an indexing structure can be produced for accessing data objects with similar features.

Previous Work

There are many methods to generate partitions which lead to indexing structures suitable for indexing in image databases, for example, R-tree [Guttman(1984)], Quadtree [Samet(1984)], general hierarchical clustering methods, and VP-tree [Yianilos(1993)]. They work fine for many situations, but all of them seem to fail to retrieve similar database objects when a nearest neighbor query lies on the partition boundary. It is because some of the similar database objects may be clustered in another partition.

We propose to use *competitive learning* clustering [Rumelhart and Zipser(1985), Haykin(1994)] algorithm to produce indexing structures for efficient and accurate information retrieval in image databases. It gives better results for the boundary queries of nearest neighbor search than R-tree, and VP-tree because it pays attention to the input distribution of the feature vectors in order to produce natural clustering but R-tree, and VP-tree do not. On the other hand, although hierarchical clustering methods are more accurate, the competitive learning clustering method is more efficient than them since they are often computationally intensive resulting in impractical use for a large set of feature vectors.

In Section 2, we will briefly describe the competitive learning clustering method and show how to use it to solve the database object indexing problem. Then, we present the experiment results as well as some discussion in Section 3. A conclusion will be drawn in Section 4.

2 Using Competitive Learning Clustering for Image Database Indexing

We use the competitive learning clustering algorithm for image database indexing. First, we use the clustering algorithm to partition the input feature set into clusters. After the clustering procedure, we map the feature vectors in the feature space into an indexing structure for efficient nearest neighbor search. Since the competitive learning generates clusters based on natural partitions of the feature vector distribution, the boundary query problem can be handled in a better way.

2.1 Competitive Learning Clustering

In this section, we present the technique of using competitive learning for clustering. There are some basic conditions of the competitive learning rule [Rumelhart and Zipser(1985), Haykin(1994)] :

- Start with a set of neurons that are all the same except for some randomly distributed synaptic weights which makes each of them respond differently to a set of input patterns.
- Limit the "strength" of each neuron.
- Allow the neurons to compete for the right to respond to a given subset of inputs.

For a specify input pattern, the neurons compete among themselves and only one of them will win the competition which is called a *winner-takes-all* neuron. The rule will then move the synaptic weight vector of the winning neuron toward the input pattern. For image databases, the feature vectors are the input patterns. By training the neurons with the feature vectors under the competitive learning rule, the weight vectors of the neurons will become the centers of the clusters of the vectors.

Let k be the number of clusters (i.e. number of neurons) and c_i, $i = 1, 2, \ldots, n$, be the cluster center points.

- Step 0: *Initialization* Randomly pick c_i as the the initial cluster centers.
- Step 1: *Competition* Randomly take a feature vector x from the feature sample set X, the winner-takes-all neuron w is that whose cluster center (weight vector) c_w is the closet to x in the sense of L_2-norm distance (Euclidean distance), i.e.,

$$\|x - c_w\|^2 = \min_i \|x - c_i\|^2 \tag{1}$$

- Step 2: *Updating Cluster Centers* Update the cluster center c_w by

$$\Delta c_w = \begin{cases} \alpha_w(x - c_w), & \text{if neuron } w \text{ wins} \\ & \text{the competition,} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

  where $0 \leq \alpha_w \leq 1$ is the learning rate for the winner-takes-all neuron.

Step 1 and 2 are iterated until the iteration converges or the number of iterations reaches a pre-specified value. The final cluster centers are the results of the competitive learning clustering. Figure 1 demonstrates eight cluster centers generated by competitive learning algorithm for an eight Gaussian mixtures distribution.

2.2 Generating Indexing Structure from Competitive Learning Clusters

After the competitive learning clustering, we may build an indexing tree based on the clusters. 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 competitive learning clustering: (1) *hierarchical approach* and (2) *non-hierarchical approach.* The first approach clusters

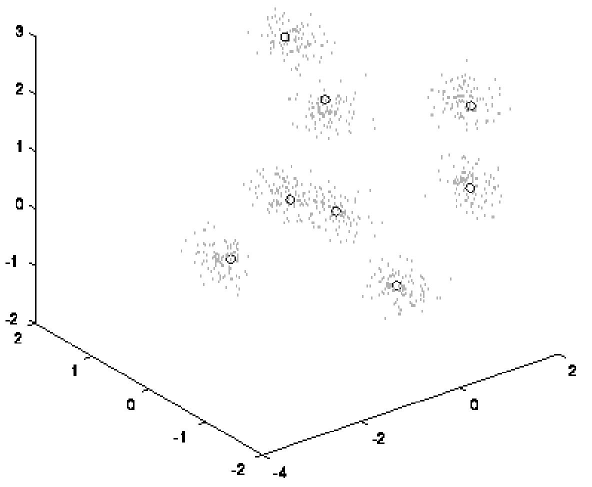

Figure 1: Competitive learning clustering with eight Gaussian mixtures with $\sigma = 0.2$. The circles indicate the cluster centers generated by the clustering algorithm.

the feature vectors which are in the subset partitioned in the previous level. The second approach considers the whole feature space each time to cluster instead. In our case, we use the non-hierarchical approach.

After the non-hierarchical top-down competitive learning clustering, there exists a mapping function that maps the clusters produced by competitive learning algorithm 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 $\hat{x}$ in L_2-norm distance 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 Experimental Results

We conducted an experiment to examine the performance of the competitive learning clustering method. We tested the method in generating the indexing structure using 2560 3-dimensional synthetic feature vectors. We used synthetic 3D feature vectors because some of the image features are 3 dimensional such as RGB color. We generated the input distribution of the feature vectors from the mixture of n Gaussian distributions $N(\mu, \sigma^2)$ where $\mu = (\mu_1, \mu_2, \ldots, \mu_n)$ and $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$. In our experiment, we simply used a constant 0.1 for σ and let $n = 2, 4, 8, 16$, and 32. For example, when $n = 8$, the synthetic feature vectors are distributed in 8 Gaussian mixtures (clusters) like Figure 1.

The experiment was conducted on an Ultra Sparc 1 machine running Matlab V4.2c. Different number of competitive learning clusters are generated for the input feature vectors. The time for competitive learning clusters is relatively short. For example, it takes about 8 seconds to generate 16 competitive learning clusters from 2560 synthetic 3D feature vectors.

Performance Measurements

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

$$\begin{aligned}\text{Recall} &= \sum_{c_i \in C \wedge c'_j \in C'} \frac{c_i \cap c'_j}{\# c_i} \\ &= \frac{\text{Number of target images retrieved}}{\text{Number of target images}} \qquad (3)\end{aligned}$$

$$\begin{aligned}\text{Precision} &= \sum_{c_i \in C \wedge c'_j \in C'} \frac{c_i \cap c'_j}{\# c'_j} \\ &= \frac{\text{Number of target images retrieved}}{\text{Number of images retrieved}} \qquad (4)\end{aligned}$$

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

We conducted 100 trails with the same set of feature vectors but with different initial starting points of the centers of the competitive learning clusters. We then calculated the average performance of the competitive learning clustering method using equations 3 and 4. Tables 1 and 2 show the results for the average Recall and Precision measurements.

Table 1: Recall Table. MG: Mixture Groups, CL: Competitive Learning Clusters.

MG / CL	4	8	16	32
2	0.5016	0.2629	0.1612	0.1221
4	0.9983	0.5411	0.3292	0.2336
8	0.9825	0.9941	0.7577	0.6134
16	0.9582	0.9493	0.8682	0.7143
32	0.9403	0.9451	0.9065	0.7790

Table 2: Precision Table. MG: Mixture Groups, CL: Competitive Learning Clusters.

MG / CL	4	8	16	32
2	1.0000	1.0000	1.0000	1.0000
4	0.9771	0.8871	0.9997	0.9751
8	0.4456	0.9510	0.9947	0.9381
16	0.2222	0.4341	0.7495	0.7847
32	0.1044	0.2137	0.3860	0.5225

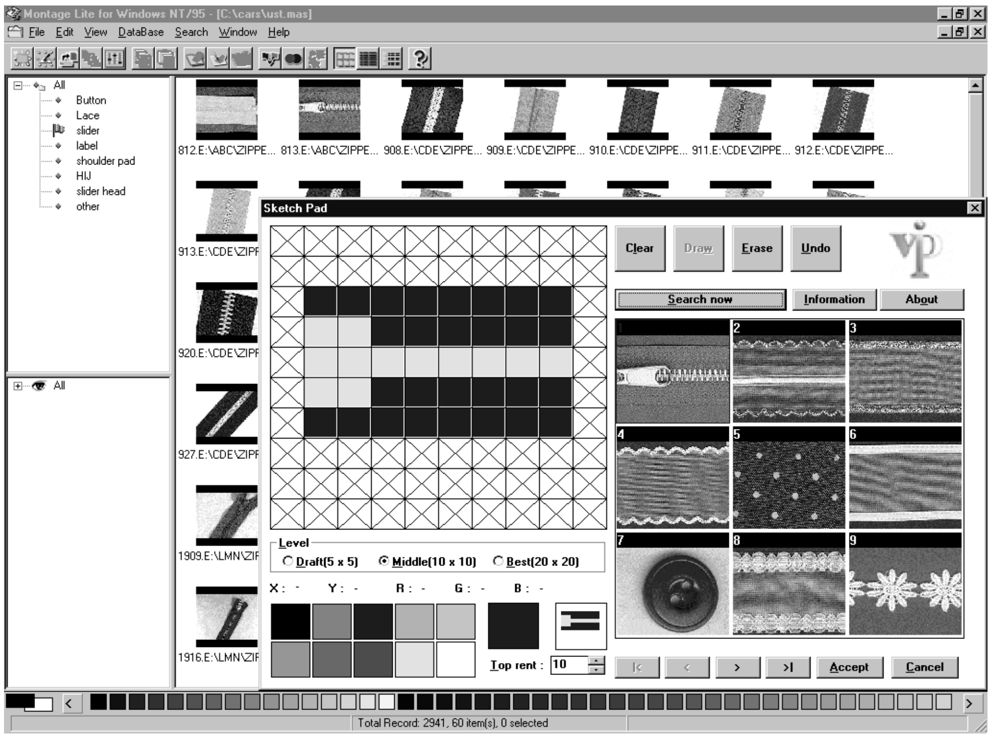

Figure 2: The user-interface of Montage. There are an image catalog in the background and a sketch pad query dialog box in the front.

Discussion

From Table 1, we find that the Recall values are relatively high when the number of Gaussian mixtures are greater than or equal to the number of competitive learning clusters. It means that the boundary query problem mentioned in Section 1 is well handled because most of the target images are retrieved with suitable competitive learning clustering.

There is a problem when the number of competitive learning clusters is greater than the number of the Gaussian mixtures. We may find in Table 1 that the Recall values are relatively low in this case. It is because multiple competitive learning 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.

Montage : An Image Database Supporting Content-based Retrieval

Montage is an image database for Hong Kong's fashion, textile, and clothing industry [King et al.(1995)]. It supports content-based retrieval using *color*, *texture*, *sketch*, and *shape*. The system is useful for many applications such as image cataloging, image editing and image browsing. Figure 2 shows the user-interface of Montage with a sketch pad query dialog.

One of the key issues in Montage is the implementation of a good indexing structure for rapid and efficient image retrieval in a very large database. Currently, Montage is using R-tree indexing for retrieval. Plan is underway to implement competitive learning indexing technique into the system for efficient and accurate content-based image retrieval.

4 Conclusion

In this paper, we use competitive learning clustering algorithm to produce a good indexing structure for efficient and accurate information retrieval in image databases. We use Recall and Precision to measure the performance of image searching based on the indexing structure generated by the algorithm. The results of the performance test show that the competitive learning clustering algorithm works fine for the boundary queries of nearest neighbor search. Currently, we plan to integrate this competitive learning indexing technique into Montage for content-based image retrieval.

Acknowledgemnets

The work was supported in part by the Hong Kong Research Grant Council # CUHK 485/95E and IDTC Grant # AF/17/95. We also thank Prof. M. Tsang of HKUST for providing some of the images shown in Figure 2.

References

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

[Haykin(1994)] S. Haykin. *“Neural networks: a comprehensive foundation”*. Macmillan, New York, 1994.

[King et al.(1995)] I. King, A. Fu, L.W. Chan, and L. Xu. “Montage: An Image Database for the Hong Kong’s Textile, Fashion, and Clothing Industry”, 1995. http://www.cse.cuhk.edu.hk/∼viplab.

[Rumelhart and Zipser(1985)] D.E. Rumelhart and D. Zipser. “Feature discovery by competitive learning”. *Cognitive Science*, 9:75–112, 1985.

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

[Yianilos(1993)] 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.