TOWARDS OPTIMAL CLUSTERING FOR APPROXIMATE SIMILARITY SEARCHING

Ertem Tuncel and Kenneth Rose

Dept. of ECE, University of California, Santa Barbara, CA 93106
{ertem,rose}@ece.ucsb.edu

ABSTRACT

We propose an iterative optimization algorithm for the generic class of clustering-based indexing for approximate similarity searching. It was previously shown that clustering is a powerful component of approximate searching that reduces the number of retrieved data points. The proposed algorithm's objective is to maximize the expected search quality given the query distribution. The problem is decomposed into minimization over three mapping functions, and the fixed-point iterations of the algorithm alternately optimize one mapping while fixing the other two. We demonstrate via experiments on real high dimensional data sets that the algorithm significantly improves the time/accuracy efficiency over heuristic clustering design.

1. INTRODUCTION

The similarity search problem is central in a wide range of applications in multimedia databases, which may contain images, video, text, music, etc. The degree of similarity between two objects is often measured by a distance function, e.g., the Euclidean distance, operating on feature vectors extracted from the data. The user submits a query object to a search engine, and may either provide a distance threshold ϵ, or the number of objects k to be returned. These types of queries are called range query, and k-nearest-neighbor (k-NN) query, respectively.

The feature vector dimension is usually very high and the search procedure is subject to Bellman's notorious "curse of dimensionality" [1], i.e., the search space grows exponentially with the number of dimensions. It was shown in [2] that all current index techniques degrade to linear search for high enough dimensionality. However, significant savings in disk I/O cost are possible if one allows for approximate search results. Usually, the extraction of feature vectors from the data objects is itself a heuristic process that attempts to approximately capture relevant information. Moreover, even if the feature vectors represent the original data with 100% accuracy, users would still differ in their perception systems, and hence in their similarity expectations. Thus, rather than incur the extremely high cost of an exact result, it is more cost-effective to develop a fast search engine that outputs an approximate set.

THIS WORK IS SUPPORTED IN PART BY THE NSF UNDER GRANTS NO. EIA-9986057 AND EIA-0080134, THE UNIVERSITY OF CALIFORNIA MICRO PROGRAM, DOLBY LABORATORIES, INC., LUCENT TECHNOLOGIES, INC., MINDSPEED TECHNOLOGIES, INC., AND QUALCOMM, INC.

Various approximate similarity search algorithms exist in the literature. For a comprehensive survey on approximate algorithms, see [3]. In this paper, we focus on a generic approach; clustering-based approximate similarity search algorithms. In a clustering-based indexing algorithm, clusters are stored in a sequential file for improved I/O efficiency. To answer queries, a few clusters (chosen by a decision process depending on the query) are retrieved into the main memory. Among the retrieved objects, those that are most similar to the query point constitute the answer set. More specifically, in a range query, the answer set consists of the retrieved objects whose distances to the query point are less than ϵ. On the other hand, in a k-NN query, the objects are sorted by their distances to the query and the best k are returned.

The power of clustering relative to other indexing techniques has been addressed by others in [4] where clustering's superiority is argued. From our perspective here, however, clustering-based indexing is a component of the system whose objective is to reduce the number of retrieved data points. It can later be combined with other methods that aim to reduce the information retrieved for each data point. It is evident that such a combination is necessary for increased time/accuracy efficiency [5].

To the best of our knowledge, there is no algorithm in the literature that measures and therefore targets the performance of clustering during the design stage. Instead, a heuristic method is usually employed (see for example [4]). To measure the performance accurately, it is of paramount importance to take into account the accumulated query history. If no such history exists, the data set itself can be used as an approximate model for that purpose. In this work, we take a step towards optimal clustering by proposing an iterative optimization algorithm that targets the expected search quality given the query distribution. The proposed algorithm can be initialized by the output of any other clustering method, and improves the performance. Finally, our approach is general and is not restricted to any specific distance function as the measure of similarity.

2. PRELIMINARIES AND NOTATION

Let $\mathcal{X}$ and $\mathcal{Y}$ denote the data and the query spaces, respectively. We assume prior knowledge of the data distribution $p_X(x)$, and query distribution $p_Y(y)$. Let the similarity between x and y be measured by $\rho : \mathcal{X} \times \mathcal{Y} \longrightarrow [0, \infty)$. Let N be the number of clusters. We denote the clustering function by $f : \mathcal{X} \longrightarrow \mathcal{I}$, where $\mathcal{I} = \{1, 2, \ldots, N\}$ is the cluster index set. The scenario we consider dictates that the decision to access a particular $x \in \mathcal{X}$ must be purely based on $f(x)$ and $y \in \mathcal{Y}$. We can accurately model the decision by a function $h : \mathcal{I} \times \mathcal{Y} \longrightarrow \{0, 1\}$.

Let $G(y) \subset \mathcal{X}$ be the golden (i.e., the true) answer set for the query y. Denote by $r(y)$ the radius of the query sphere, i.e., the distance of the farthest data point in $G(y)$ to the query point:

$$r(y) = \max_{x \in G(y)} \rho(x, y) .$$

Similarly, let $A(y) \subset \mathcal{X}$ denote the approximate answer set for y. The set $A(y)$ is obtained by retrieving all clusters i for which $h(i, y) = 1$, and choosing a subset of the retrieved points according to the search criterion.

We denote by $T(y)$ the fraction of accessed data points, i.e.,

$$T(y) = \int_{\mathcal{X}} p_X(x) 1[h(f(x), y)] dx ,$$

where $1[\cdot]$ is the indicator function. The data points in each cluster are stored sequentially on disk, and each time a query is processed, very few clusters are retrieved from disk. Hence, during the processing of a query $y \in \mathcal{Y}$, very few random seeks are needed, and the total I/O time expended is approximately proportional to the number of accessed data points, $T(y)$.

The accuracy of an approximate similarity search is usually measured by the achieved recall, i.e.,

$$\text{Recall}(y) = \frac{|A(y) \cap G(y)|}{|G(y)|} .$$

A higher recall means a more accurate approximate answer. We further assume that if a data point $x \in G(y)$ is accessed, then it must fall into $A(y)$, that is, a point in the golden set is falsely dismissed if and only if it is not accessed. This is a fair assumption, since it is true for both range and k-NN queries, which are the most popular similarity searching schemes. An equivalent expression for the recall then becomes

$$\text{Recall}(y) = 1 - \int_{\mathcal{X}} p_X(x) d(x, y, h(f(x), y)) dx ,$$

where $d : \mathcal{X} \times \mathcal{Y} \times \{0, 1\} \longrightarrow [0, \infty)$ is defined as

$$d(x, y, z) = \begin{cases} \frac{1}{|G(y)|} & x \in G(y) \text{ and } z = 0 \\ 0 & \text{otherwise} \end{cases} .$$

In this paper, we also consider an alternative accuracy measure $D(y)$ given by

$$D(y) = 1 - \int_{\mathcal{X}} p_X(x) d(x, y, h(f(x), y)) dx ,$$

where

$$d(x, y, z) = \begin{cases} \frac{1}{|G(y)|} \left[1 - \frac{\rho(x,y)}{r(y)}\right] & x \in G(y) \text{ and } z = 0 \\ 0 & \text{otherwise} \end{cases} .$$

We call this measure the distance-sensitive recall, since it penalizes false dismissal of data close to the query point more severely than that of data near the boundary of the query sphere.

3. CLUSTERING DESIGN

An optimal time/accuracy trade-off is achieved by the following minimization:

$$\min_{f(\cdot), h(\cdot,\cdot)} \int_{\mathcal{Y}} p_Y(y) T(y) dy ,$$

subject to $\int_{\mathcal{Y}} p_Y(y) D(y) dy \geq D$. Equivalently, we can attack the Lagrangian cost

$$L = \int_{\mathcal{Y}} p_Y(y) \left[T(y) - \lambda D(y)\right] dy , \qquad (1)$$

where $\lambda > 0$ is a fixed multiplier. Before proceeding further, let us further simplify (1) by introducing

$$l_\lambda(x, y, z) = \begin{cases} \frac{\lambda}{|G(y)|} \left[1 - \frac{\rho(x,y)}{r(y)}\right] & x \in G(y) \text{ and } z = 0 \\ 1 & z = 1 \\ 0 & \text{otherwise} \end{cases} .$$

We can now equivalently minimize

$$L = \int_{\mathcal{Y}} \int_{\mathcal{X}} p_X(x) p_Y(y) l_\lambda(x, y, h(f(x), y)) dx dy . \qquad (2)$$

Note, however, that $h(i, y)$ should be further constrained in the above minimization for practicality purposes. In reality, the only knowledge we have about $p_Y(y)$ comes from a training set $\mathcal{Q}$ of limited size, so we can only design $h(i, y)$ for $y \in \mathcal{Q}$. As an approximation, for all other points in the query space, we can use the nearest $y \in \mathcal{Q}$ to evaluate $h(i, y)$, but this requires a huge memory, since the values of $h(i, y)$ must be stored in a table of size $N \times |\mathcal{Q}|$.

The simplest approach, also adopted in [4], is to constrain $h(i, y)$ to be solely based on the distance $\rho(\hat{x}_i, y)$, where $\hat{x}_i$ is the centroid of cluster i. Essentially, cluster centroids are sorted in increasing order of $\rho(\hat{x}_i, y)$, and only few clusters corresponding to the nearest $\hat{x}_i$ are accessed. This special form of $h(i, y)$ makes it extremely difficult to attack (2), because for fixed $f(x)$ the Lagrangian cost function becomes piecewise constant in $\{\hat{x}_i\}_{i=1}^{N}$. Perhaps the only remedy to this difficulty is to cluster the data set with a heuristic method, and to extract the corresponding cluster centroids, as done in [4]. In fact, for comparison purposes, we implemented such a simple scheme by clustering the data set with the K-means algorithm [6]. Note that this approach is overly simplistic in that the knowledge of the query distribution $p_Y(y)$ is not exploited.

Our approach is to constrain $h(i, y)$ to be of the form $h'(i, g(y))$, where $g : \mathcal{Y} \longrightarrow \mathcal{J} = \{1, 2, \ldots, M\}$, is a partitioning of the query space. Note that it suffices to store in memory an $N \times M$ table for the values $h'(i, j)$. However, now $g(y)$ has to be in a special form that facilitates its evaluation outside the query training set $\mathcal{Q}$. For that purpose we constrain $g(y)$ to be a nearest-neighbor partitioning, i.e.,

$$g(y) = \arg\min_{j \in \mathcal{J}} \delta(y, \hat{y}_j) ,$$

where $\hat{y}_j \in \mathcal{Y}$ are the partition centroids, and $\delta : \mathcal{Y} \times \mathcal{Y} \longrightarrow [0, \infty)$ is a distance measure in the query space. Note that the domain of δ is in general different from that of ρ, and even when $\mathcal{X} = \mathcal{Y}$, we do not assume $\rho(\cdot,\cdot) = \delta(\cdot,\cdot)$. The resultant Lagrangian minimization problem

$$\min_{f,g,h'} \int_{\mathcal{Y}} \int_{\mathcal{X}} p_X(x) p_Y(y) l_\lambda(x, y, h'(f(x), g(y))) dx dy \quad (3)$$

is still difficult to attack in terms of $\{\hat{y}_j\}_{j=1}^{M}$. To overcome this difficulty, we could fix $\{\hat{y}_j\}_{j=1}^{M}$ (and therefore $g(y)$), by first running a clustering algorithm on the query training set $\mathcal{Q}$, and then minimizing the Lagrangian with respect to $f(x)$ and $h'(i,j)$ only. But, this approach could result in a very degraded performance, since it completely ignores the data distribution during the design of $g(y)$. Instead, we initially neglect the nearest-neighbor partitioning assumption on $g(y)$, and attack (3). In effect, we design clustering of the data and the query spaces simultaneously, together with the decision function. In a second round, we compute the centroids $\{\hat{y}_j\}_{j=1}^{M}$, fix $g(y)$ as the corresponding nearest-neighbor partitioning, and then re-optimize the Lagrangian cost function with respect to $f(x)$ and $h'(i,j)$.

The design algorithm we propose is a fixed-point optimization method, which is usually encountered as a powerful tool in various quantization problems [7]. The principle is to fix two of the mappings $f(x)$, $g(y)$, and $h'(i,j)$, and optimize the other. We proceed by giving explicit expressions for the optimal mappings:

- For fixed $f(x)$ and $g(y)$, minimize

$$L_{i,j} = \int_{\mathcal{X}_i} \int_{\mathcal{Y}_j} p_X(x) p_Y(y) l_\lambda(x, y, h'(i,j)) dx dy$$

over $h'(i,j)$, where $\mathcal{X}_i = \{x : f(x) = i\}$, and $\mathcal{Y}_j = \{y : g(y) = j\}$. The optimal $h'(i,j)$ is determined by

$$h'(i,j) = \arg \min_{z \in \{0,1\}} L_{i,j}^{z} ,$$

where

$$L_{i,j}^{0} = \lambda \int_{\mathcal{Y}_j} \int_{\mathcal{X}_i \cap G(y)} \frac{p_X(x) p_Y(y)}{|G(y)|} \left[1 - \frac{\rho(x,y)}{r(y)}\right] dx dy ,$$

and $L_{i,j}^{1} = |\mathcal{X}_i| \cdot |\mathcal{Y}_j|$.

- For fixed $g(y)$ and $h'(i,j)$, the optimal $f(x)$ is given by

$$f(x) = \arg \min_{i \in \mathcal{I}} \int_{\mathcal{Y}} p_Y(y) l_\lambda(x, y, h'(i, g(y))) dy .$$

- For fixed $f(x)$ and $h'(i,j)$, the optimal $g(y)$ is given by

$$g(y) = \arg \min_{j \in \mathcal{J}} \int_{\mathcal{X}} p_X(x) l_\lambda(x, y, h'(f(x), j)) dx .$$

We can initialize the algorithm by fixing any two of $f(x)$, $g(y)$, and $h'(i,j)$. We choose to fix $f(x)$ and $h'(i,j)$. Assume that $f(x)$ is given to us by a competent clustering algorithm. We first produce the centroids $\{\hat{x}_i\}_{i=1}^{N}$. Denote by $n_1(y)$ and $n_2(y)$ the indices of the nearest two centroids for $y \in \mathcal{Q}$. For each y, we then add $n_1(y)$ and $n_2(y)$ to the list of encountered pairs so far. After we visit all $y \in \mathcal{Q}$, we associate query clusters with the elements in the list. If the jth pair in the list is (n_1, n_2), we set $h'(n_1, j) = h'(n_2, j) = 1$, and set the rest of the $h'(i,j)$ values to 0.

4. EXPERIMENTAL RESULTS

In this section, we demonstrate the achieved improvement of our algorithm over K-means clustering. We performed experiments on two real world data sets. The first one is a 64-dimensional color histogram set, and the second one consists of 60-dimensional texture features of an aerial photo data set. Both sets are of size 10,000. We assume that $\rho(x,y) = \delta(x,y) = \|x - y\|^2$. We randomly pick a 2,500-element subset of the data set as our query training set, and test the performance of compared clustering algorithms over another disjoint subset of size 1,000. We partition both data sets into 32 clusters.

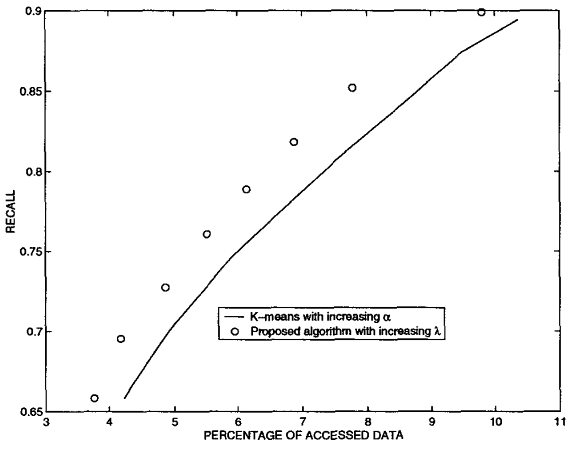

Figure 1: Comparison of recall versus percentage of accessed data for 50-NN searching on the Histogram data set.

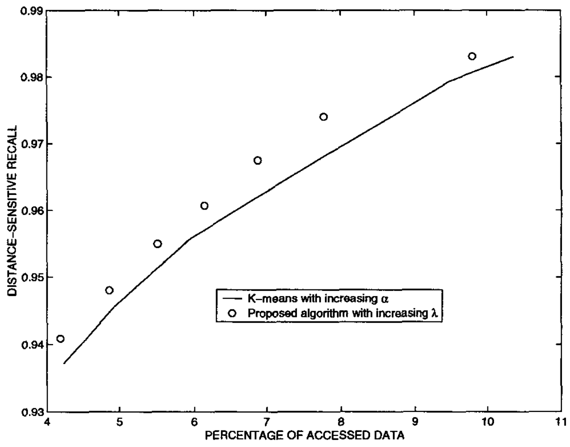

Figure 2: Comparison of distance-sensitive recall versus percentage of accessed data for 50-NN searching on the Histogram data set.

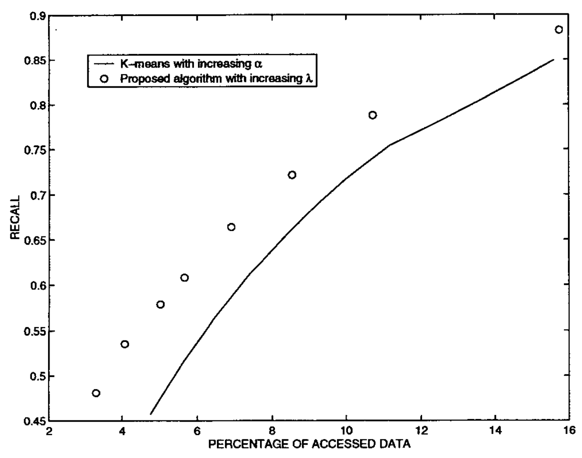

Figure 3: Comparison of recall versus percentage of accessed data for 100-NN searching on the Aerial photo data set.

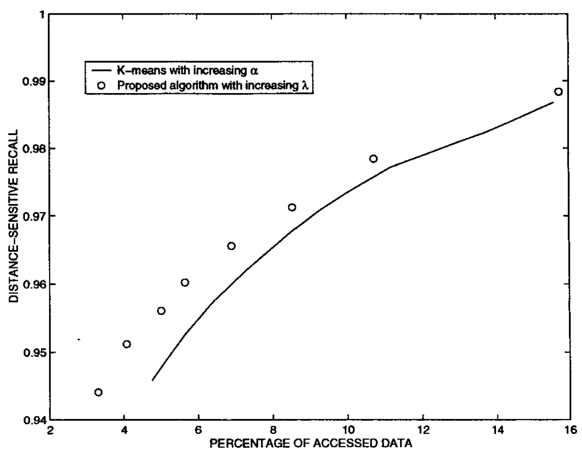

Figure 4: Comparison of distance-sensitive recall versus percentage of accessed data for 100-NN searching on the Aerial photo data set.

For the K-means clustering, the decision function $h(i, y)$ for k-NN search is determined as follows: Fix some $\alpha > 1$. Sort $\rho(\hat{x}_i, y)$ in increasing order. Retrieve clusters in order until the total number of retrieved objects exceeds $\alpha \times k$.

In Figures 1, 2, 3, 4, we compare the time/accuracy performance of the K-means algorithm and the proposed algorithm. Although the proposed algorithm targets optimal trade-off between distance-sensitive recall and I/O time, we also present the achieved recall versus I/O time plots. The solid lines show the results for the K-means algorithm with changing α. As α increases, so does the I/O time complexity and the accuracy. The circles reflect the performance of the proposed algorithm with changing λ. The achieved points are not connected with a line to emphasize the fact that they are produced by separate runs of the algorithm with different λ. Increasing λ puts more weight on the accuracy component of the Lagrangian (3), and hence yields better accuracy with the expense of worse time complexity.

Note that both the clustering schemes yield better performance on the Histogram data set. That is because the Histogram set is of clustered nature, whereas the Aerial photo set is somewhat more uniformly distributed. This phenomenon was also observed in [4]. Another observation is that by looking at the achieved recall only, the performance could be considered poor when the time complexity is kept low (say, only around 4% of the whole data set is accessed). However, the achieved distance-sensitive recall is over 93%, which shows that most of the falsely dismissed points are in fact close to the boundary of the query sphere. For example, when the recall is 0.5, and the distance-sensitive recall is 0.95 for 100-NN searching, as is the case for the Aerial photo set, by easy algebra we can see that the average distance of the falsely dismissed points to the query point y is around $0.9r(y)$.

When we compare the I/O time complexity for fixed values of recall or distance-sensitive recall, we observe a speedup of as high as 1.125 for the Histogram data set, and 1.54 for the Aerial photo data set.

5. REFERENCES

[1] R. Bellman, Adaptive Control Processes: A Guided Tour, Princeton University Press, 1961.

[2] R. Weber, H.-J. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," in Proceedings of the International Conference on Very Large Databases (VLDB), New York City, New York, August 1998, pp. 194–205.

[3] P. Ciaccia and M. Patella, "Approximate similarity queries: A survey," unpublished technical report. See www-db.deis.unibo.it/research/papers/TRs/CSITE-08-01.pdf.

[4] C. Li, E. Chang, H. Garcia-Molina, and G. Wiederhold, "Clustering for approximate similarity search in high-dimensional spaces," to appear in IEEE Transactions on Knowledge and Data Engineering.

[5] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, and A. El Abbadi, "Approximate nearest neighbor searching in multimedia databases," in Proceedings of 17th IEEE International Conference on Data Engineering (ICDE), Heidelberg, Germany, April 2001, pp. 503–511.

[6] R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.

[7] A. Gersho and R. M. Gray, Vector Quantization and Signal Compression, Kluwer Academic Publishers, Boston, MA, 1992.