Clustering and Singular Value Decomposition for Approximate Indexing in High Dimensional Spaces [*]

Alexander Thomasian [†], Vittorio Castelli, and Chung-Sheng Li
IBM Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598
Telephone: (203) 251-9516, (914) 784-7665 (914) 784-6661
Email: {athomas,vittorio,csli}@watson.ibm.com

Abstract

High-dimensionality indexing of feature spaces is critical for many data-intensive applications such as content-based retrieval of images or video from multimedia databases and similarity retrieval of patterns in data mining. Unfortunately, the performance of nearest neighbor (NN) queries, which are required for similarity search, deteriorates rapidly with the increase in the number of dimensions. We propose the Clustering with Singular Value Decomposition (CSVD) method, which combines clustering and singular value decomposition (SVD) to reduce the number of index dimensions, while maintaining a reasonably high precision for a given value of recall. In the proposed CSVD method, homogeneous points are grouped into clusters such that the points in each cluster are more amenable to dimensionality reduction than the original dataset. Experiments with texture vectors extracted from satellite images show that CSVD achieves significantly higher dimensionality reduction than SVD for the same fraction of total variance preserved. Conversely, for the same compression ratio CSVD results in an increase in preserved total variance with respect to SVD (e.g., a 70% increase for a 20:1 compression ratio). This translates to a higher efficiency in processing approximate NN queries, as quantified through experimental results.

1 Introduction

Similarity-based retrieval from a large collection of vectors with high- dimensionality has become increasingly important for indexing multimedia databases. In particular, there has been a proliferation of image and video databases that allow content-based retrieval of images and videos by using low-level features such as texture, color histogram, and shape in diverse areas such as photographic images, medical images, video clips, and satellite images. Efficient similarity search invariably requires precomputed features. The number of dimensions of the feature vectors can be potentially large. Similarity search is then equivalent to a *nearest neighbor* (NN) query in the high- dimensionality space of the feature vectors. NN search based on sequential scan of large files, or tables, of feature vectors is computationally too expensive and has long response time.

Multidimensional indexing structures, which have been investigated intensively in recent years, seem to be the obvious solution to this problem. Unfortunately, as a result of the "curse of dimensionality" the efficiency of indexing structures deteriorates rapidly as the number of dimensions increases [3, 4, 9]. Even the highly-optimized X-trees [2] are inefficient for nearest-neighbor queries [3], and parallel processing solutions have been proposed to mitigate the problem [4].

In essence, the existing indexing structures perform very well in low-dimensionality spaces, and poorly in high- dimensionality spaces. A potential solution is therefore to reduce the dimensionality of the search space before indexing the data. The challenge is to achieve index space compression (which in general results in loss of information) without affecting information retrieval performance.

Dimensionality reduction methods are based on either linear transformations followed by the selection of a subset of features, although nonlinear transformations are also possible. Techniques based on linear transformations, such as the Karhunen-Loeve (KL) transform, the Singular Value Decomposition (SVD) method, and Principal Component Analysis (PCA) [10, 8], have been widely used for dimensionality reduction and data compression [12] and form the basis of our study.

SVD generally relies on "global" information derived from all the vectors in the dataset, which is more effective for datasets consisting of homogeneously distributed vectors (such as in [12]). For databases with heterogeneously distributed vectors, more efficient representation can be generated by subdividing the vectors into groups, with each of which being characterized by a different set of statistical parameters.

Based on this insight, a new method, *Clustering Singular Value Decomposition (CSVD)*, for reducing the dimensionality of the feature space is proposed in this paper. This method consists of two steps: partitioning the data set using a clustering technique [7] (see Section 4) and the application of SVD to the vectors in each cluster to produce a vector space of transformed features with reduced dimensionality. An iterative procedure then discovers the "optimal" degree of clustering based on either the average tolerable error or the data compression ratio by searching.

Experiments with texture vectors from satellite images show that CSVD preserves from 37% to 62% more vari-

[*]This paper is funded in part by grant no. NASA/CAN NCC5-101.

[†]Current affiliation is CS&E Dept. at Univ. of Connecticut, e-mail: athomas@uconnvm.uconn.edu.

ance than SVD for a 20-fold dimensionality reduction. The increase in the fraction of preserved total variance attained with CSVD yields an improvement in retrieval efficiency over systems based on SVD. Conversely, when the reduction in variance is kept at a constant 40%, SVD is shown to require, on the average, twice as many dimensions as CSVD. From the information retrieval viewpoint, experiments using the proposed CSVD indices demonstate that for a given value of *recall* the *precision* metric shows only a moderate degradation for NN queries

The rest of the paper is organized as follows. Section 2 contains the preliminary material on multidimensional indexing methods, and the mathematics behind SVD and PCA. The CSVD algorithm and the processing of NN queries in the new environment is described in Section 3. Section 4 describes the experimental results. Conclusions are given in Section 5.

2 Preliminary

2.1 Indexing Methods for Nearest Neighbor Search

In similarity search an object is mapped into a point (equivalently, a vector) in a high-dimensional feature space. The similarity between two objects U and V is then measured by a function of the "distance" between the corresponding points $\mathbf{u} = [u_1, ..., u_N]^T$ and $\mathbf{v} = [v_1, ..., v_N]^T$. The k NN query retrieves from a database the k most similar entries to a specified query object. When all the features (i.e., the entries of the vectors) are numerical, the most commonly used similarity measure is the Euclidean distance

$$D(\mathbf{u},\mathbf{v}) = \left[(\mathbf{u}-\mathbf{v})^T(\mathbf{u}-\mathbf{v})\right]^{1/2} = \sqrt{\sum_{i=1}^{n}(u_i - v_i)^2}. \quad (1)$$

Indexing of feature vectors seems to be the obvious solution, and there are many multidimensional indexing structures such as grid files, k-d-b trees, R-trees, and R*-trees to choose from [8]. However, even X-trees [2], which in general provide significant performance gains over R*-trees, do not improve the execution time of NN queries [3]. The inefficiency associated with high-dimensionality indices has been recently discussed in the context of R-tree-like structures, because of their popularity [5, 4], although similar arguments appear in [6]. In effect, as the dimensionality grows, almost all of the pages in the index need to be touched.

The method proposed in this paper can be used in conjunction with various indexing structures, especially R-trees for which efficient NN search methods exist [15]. The indexing structure described in [11] for the efficient processing of NN-queries is however utilized in our studies (not reported here due to space limitations).

The efficiency of approximate indexing is quantified by the *recall* and *precision* metrics in the information retrieval literature. For each query object X, let $A(X)$ denote the subset of the database containing the k most similar objects to X. Since the search algorithm is approximate, we request more than k NNs and let $B(X)$ denote the result set. Finally, let $C(X) = A(X) \bigcap B(X)$. Then,

- *Recall* (R) is the fraction of retrieved objects $B(X)$ that are among the k NNs of X, namely
$$R = |C(X)| \,/\, |A(X)|. \quad (2)$$
- *Precision* (P) is the fraction of the k-nearest neighbors $A(X)$ of X that are retrieved by the algorithm, i.e., $P = |C(X)| \,/\, |B(X)|$.

The higher the recall requirement the lower the precision and vice-versa.

2.2 Singular Value Decomposition

In this subsection, the well-known singular value decomposition (SVD) method is reviewed in order to establish the necessary notations for later sections.

Consider a $M \times N$ matrix $\mathbf{X}$ the columns of which are correlated. Let $\overline{\mu}$ be the column mean of $\mathbf{X}$,

$$\overline{\mu}_j = (1/M)\sum_{i=1}^{M} x_{i,j},\ 1 \le j \le N, \quad (3)$$

and let $\mathbf{1}_M$ be a column vector of length M with all elements equal to 1. SVD expresses $\mathbf{X} - \mathbf{1}_M\overline{\mu}^T$ as a product of a $M \times N$ column orthonormal matrix $\mathbf{U}$ (i.e., $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix), a $M \times M$ diagonal matrix $\mathbf{S}$ containing the singular values, and a $M \times M$ real unitary matrix $\mathbf{V}$ [14]:

$$\mathbf{X} - \mathbf{1}_M\overline{\mu}^T = \mathbf{U}\mathbf{S}\mathbf{V}^T \quad (4)$$

The columns of the matrix $\mathbf{V}$ are the eigenvalues of the covariance matrix $\mathbf{C}$ of $\mathbf{X}$, defined as

$$\mathbf{C} = \frac{1}{M}\mathbf{X}^T\mathbf{X} - \overline{\mu}^T\overline{\mu} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T. \quad (5)$$

The matrix $\mathbf{C}$ is positive-semidefinite, hence it has N non-negative eigenvalues and N orthonormal eigenvectors.

Without loss of generality, let the eigenvalues of $\mathbf{C}$ be ordered in decreasing order, i.e., $\lambda_1 \ge \lambda_2 \ge ... \ge \lambda_N$. The trace ($\Sigma$) of $\mathbf{C}$ remains invariant under rotation, i.e., $\Sigma = \sum_{j=1}^{N}\sigma_j^2 = \sum_{j=1}^{N}\lambda_j$. The fraction of energy associated with the j^{th} eigenvalue is λ_j/Σ.

The singular values are related to the eigenvalues by: $\lambda_j = s_j^2/M$ or $s_j = \sqrt{M\lambda_j}$. The eigenvectors constitute the *principal components* of $\mathbf{X}$, hence the transformation $\mathbf{Y} = \mathbf{V}\mathbf{X}$ yields uncorrelated features. PCA selects features with the highest eigenvalues. Consequently for a given p it maximizes the *cumulative percentage of the total variance* or the *fraction of total variance preserved*, which is denoted by F_{var}. Several heuristics to select p are given in [Joll86], e.g., retain transformed features whose eigenvalues are larger than the average Σ/M. This may be unsatisfactory, e.g., when an eigenvalue slightly less than Σ/M is omitted. In this paper p is selected so that F_{var} exceeds a threshold, which yields acceptable efficiency in approximate searching.

The p columns of the $\mathbf{Y} = \mathbf{X}\mathbf{V}$ matrix corresponding to the highest eigenvalues are retained, the remaining discarded. This is tantamount to projecting the vectors in $\mathbf{Y}$ onto the k-dimensional hyperplane passing through $\overline{\mu}$ and spanned by the eigenvectors of $\mathbf{C}$ corresponding to the largest eigenvalues, and representing the projected vector in the reference system having as origin $\overline{\mu}$ and as coordinate axes the eigenvectors of $\mathbf{C}$.

3 The CSVD Algorithm

3.1 Preprocessing of Dataset

Before constructing the index care should be taken to appropriately scale the numerical values of different features constituting the index. In the case of texture features (used as the test dataset for this paper) some features have a small range (between 2 and 3 for the fractal dimension), while others can vary significantly (the dynamic range of the gray-level histogram can easily exceed 10^5).

In this paper, we assume that all the features contribute equally to determine the similarity between different vectors. Consequently, the columns of the tables are studentized (i.e., subtract the mean and divide by the standard deviation) to obtain columns with zero-means and unit-variances:

$$x_{i,j} \leftarrow (x_{i,j} - \mu_j)/\sigma_j, \qquad 1 \leq j \leq N \text{ and } 1 \leq i \leq M. \quad (6)$$

Note that the studentization is done only once, on the original table (X), and it is not repeated after the clustering and dimensionality reduction steps described in the following section.

3.2 Proposed Algorithm

The basic steps of the CSVD index construction are partitioning, dimensionality reduction, and within-cluster index construction, which is omitted here.

1. *The partitioning step* takes $\mathbf{X}$ as its input and divides them into K homogeneous groups or clusters $\mathbf{X}_k$, $1, \leq k \leq K$. The number of clusters can be a parameter or can be determined dynamically. In the current implementation, classical clustering methods such as LBG [13] or K-means [7] are used. These are vector-quantization-style algorithms [13], that generate clusters containing vectors which are similar to each other in the Euclidean metric sense.
The following step is carried out for $\mathbf{X}_k$, $1 \leq k \leq K$, but in what follows this subscript is eliminated to simplify the notation.

2. *The dimensionality reduction step* takes as input a group of vectors $\mathbf{X}$ and performs PCA to reduce the dimensionality. Given the fraction of the total variance (F_{var}) to be preserved. As described in Section 2.2, the algorithm computes the eigenvector matrix $\mathbf{V}$ and the eigenvalues $\lambda_1, \ldots, \geq \lambda_N$, in decreasing order of magnitude. The smallest number p of eigenvalues capturing at least F_{var} of the total variance is determined as $p = argmin_k \sum_{i=1}^{k} \lambda_i \geq F_{var} \cdot \sum_{i=1}^{N} \lambda_i$. The rotation $\mathbf{Y} = (\mathbf{X} - \overline{\mu})\mathbf{V}$ produces uncorrelated features, the last $N - p$ of which are discarded, i.e., set to zero.

The goal of minimizing the size of the resulting index while maximizing the speed of the search simultaneously are somewhat contradictory. Minimizing the size of the index by aggressive clustering and dimensionality reduction is very beneficial when the database is very large. On the other hand, aggressive clustering and dimensionality reduction can also force the search algorithm to visit more clusters, thus reducing the benefits of computing the distances with fewer dimensions.

Since the search is approximate, different index configurations should be compared in terms of recall and precision (see Section 2.1). These two metrics are difficult to incorporate into an optimization procedure, because their values can only be determined experimentally (see Section 4).

An optimization algorithm is required to select the number of clusters and the number of dimensions to be retained for given constraints on precision and recall.

This problem can be solved in two steps by noting that both recall and precision are functions of F_{var}. Hence an index that maximizes F_{var} can be designed for a given data compression objective. If the value of F_{var} is small, the optimization with a lower data compression objective is repeated. The index is then verified to confirm whether it yields satisfactory precision and recall. This is done by constructing a test set, searching for the k NNs of its elements using an exact method and the approximate index, thus estimating precision and recall. If the performance is not adequate, the input parameters for designing the index are respecified. Including this last step as part of the optimization is possible, but would have made the initial index design step very lengthy.

A direct consequence of the dimensionality reduction of the index is the reduction of its size. The size of the index is affected by the efficiency of data representation and the size of the clustering and SVD information, namely, the $\mathbf{V}$ matrices for coordinate transformation and the bit-vectors specifying the principal components selected by SVD. For large tables and a non-aggressive clustering strategy, the clustering and SVD information are a negligible fraction of the overall data. Thus, we define the *index volume* as the total number of entries preserved by the index. The original volume is $V_0 = M \times N$, while given K clusters with m_k points and p_k dimensions in cluster k, the new volume is $V_1 = \sum_{k=1}^{K} m_k p_k$. The fraction of volume retained is then $F_{vol} = V_1/V_0$ and the mean number of dimensions is $F_{vol}N$.

A procedure which determines an "optimal" degree of clustering (K) for a given F_{vol} so as to maximize F_{var} is described below. First note that the reduction in the volume objective directly translates to the mean number of dimensions in the resulting index: $\overline{p}(F_{vol}) = V_1/M$.

Consider increasing values of K and for each K use a binary search to determine $F_{var}(K)$, which leads to $\overline{p}(F_{vol})$, which is in the range $V_1/M \pm 0.5$. A point of diminishing returns is reached as K increases, and the search is terminated when $F_{var}(K+\delta)/F_{var}(K) \leq 1+\epsilon$ (e.g., $\epsilon = 0.01$ and trivially $\delta = 1$). In effect, although F_{var}^{max} is not achieved, the smaller K will result in a smaller index size when the size of the meta-data file is also taken into account.

The maximum value for F_{var} (F_{var}^{max}) for the input parameter F_{vol} varies with the dataset. A relatively reasonable F_{vol} (based on experience with similar datasets) may result in an F_{var}^{max} which is too small to sustain efficient processing of NN queries. The user can specify a larger value for F_{vol} to ensure a higher F_{var}^{max}. Alternatively, a smaller F_{vol} may be considered if F_{var}^{max} is too high.

After the index is created additional tests are required to ensure that the index provides a sufficient precision for a given recall. The query points may correspond to randomly selected vectors from the input table. The index is unsatisfactory if the mean precision ($\overline{P}$) over a set of samples is smaller than a pre-specified threshold.

3.3 Searching for the Nearest Neighbors

Once the index is generated, it can be used for both exact and similarity searches. The first operation is the initial step required by the second one.

In an exact search all the element of the table that are equal to a query point $\mathbf{T}$ must be retrieved. In response to an exact search, $\mathbf{T}$ is first studentized with the same coefficients used for the entire database, thus yielding $\mathbf{T}'$. If a first rotation or dimensionality reduction step was applied to construct the index, and in this case $\tilde{\mathbf{T}}'' = \mathbf{T}'\mathbf{V}$ is computed and all the discarded dimensions are set to zero to produce $\mathbf{T}''$ (recall that the dataset is studentized, thus $\overline{\mu}$ is now the origin of the Euclidean coordinate system). The cluster i to which $\mathbf{T}''$ belongs, referred to as the *primary cluster*, is then found. The actual details of this step depend on the selected clustering algorithm. For LBG, which produces an optimal tessellation of the space given the centroid selection, $i = \mathrm{argmin}_j \left\{D_2(\mathbf{T}'', \overline{\mu}_j)\right\}$, (where $D_2(\cdot,\cdot)$ is the Euclidean distance,) i.e., $\mathbf{T}''$ belongs to the cluster with the closest centroid, thus, the computational cost is $O(K)$, where again K is the number of clusters. For tree-structured vector quantizers (TSVQ) the labeling step is accomplished by following a path from the root of a tree to one of its leaves, and if the tree is balanced, the computational cost is $O(\log K)$. Once the cluster is determined, $\mathbf{T}''$ is projected onto the corresponding subspace. As before, $\tilde{\mathbf{T}}''' = (\mathbf{T}'' - \overline{\mu}_i)\mathbf{V}_i$ is computed, and only the relevant dimensions are retained, thus yielding $\mathbf{T}'''$. Finally, the within-cluster index is used to retrieve all the records that are equal to $\mathbf{T}'''$. By construction, the CSVD index does result in false hits, but not in misses. False hits can be filtered in a post-processing phase, by comparing $\mathbf{T}$ with the originals versions of the retrieved records.

The emphasis in this paper is on similarity searches. In particular, most of the similarity retrievals involves the retrieval of the k NNs of the query point $\mathbf{T}$ from the database. The first step of a similarity search consists of the operations described in the exact search context. After the query point is projected onto the subspace of the primary cluster, a k-nearest neighbor search is issued using the within-cluster index. Unlike the exact case, though, the search cannot be limited to just one cluster. If the query point falls near the boundaries of two or more clusters (which is a likely event in high dimensionality spaces), some of its k NNs can belong to some of the neighboring clusters. It is clearly undesirable to search all the remaining clusters, since only a few of them are *candidates*, while the others cannot possibly contain any of the neighbors of $\mathbf{T}$. To determine the candidates the simple strategy illustrated in Figure 1 is used. After searching the primary cluster, k points are retrieved. The distance between $\mathbf{T}$ and the farthest retrieved record is an upper bound to the distance between $\mathbf{T}$ and its kth neighbor $\mathbf{N}_k$. A cluster is not a candidate, and thus can be discarded, if all its points are farther away from $\mathbf{T}$ than $\mathbf{N}_k$. To identify non-candidate clusters, the cluster boundaries are approximated by minimum bounding spheres. The distance of the *farthest* point in cluster C_i from its centroid is referred to as its radius, is denoted by R_i, and is computed during the construction of the index.

A cluster is discarded if the distance between its hypersphere and $\mathbf{T}$ is greater than $D_2(\mathbf{T}, \mathbf{N}_k)$, and considered a candidate otherwise. Note, incidentally, that $\mathbf{T}$ can belong to the hyper-sphere of a cluster (such as, but not necessarily only, the primary cluster), in which case the distance is by definition equal to zero. The distances between $\mathbf{T}$ and the hyper-spheres of the clusters are computed only once. The candidate cluster (if any) with hyper-sphere closer to $\mathbf{T}$ is visited first. Ties are broken by considering the distances between $\mathbf{T}$ and centroids, as is the case, for instance, if $\mathbf{T}$ belongs to the hyper-spheres of several clusters. In the unlikely case that further ties occur (i.e., $\mathbf{T}$ is equidistant from the hyper-spheres and the centroids of two clusters), the cluster with smaller index is visited first. If the current NNs list is updated during the within-cluster search, $\mathbf{N}_k$ changes and $D_2(\mathbf{T}, \mathbf{N}_k)$ is reduced. Thus, the list of non-visited candidates must be updated. The search process terminates when the candidate list is empty.

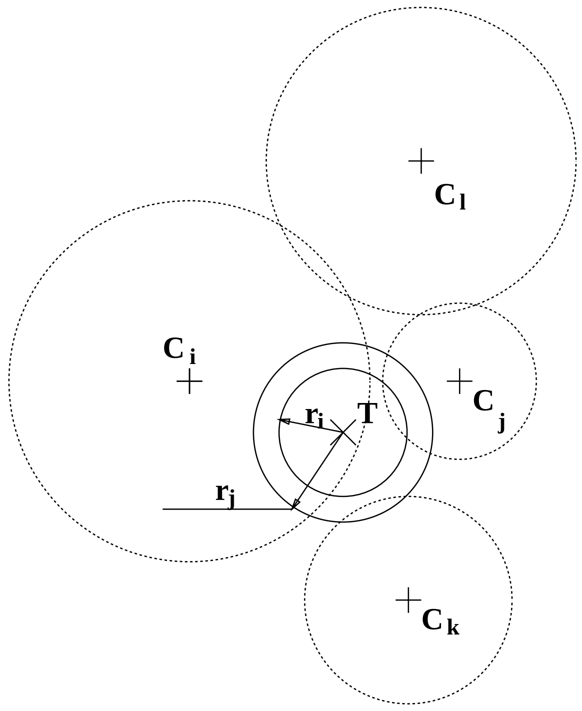

Figure 1: Searching nearest neighbors across multiple clusters.

An alternative, simpler, strategy is outlined below. Denote the distance between the centroids of two clusters C_i and C_j by $d_{i,j}$, and define the distance between *two clusters* as the distance between their hyper-spheres, i.e., $d_{i,j} - R_i - R_j$. While $d_{i,j} > R_i + R_j$ is true in some cases, $d_{i,j} < R_i + R_j$ is also possible. During the search phase, the clusters that intersect the primary cluster are checked in increasing order of distance, while clusters with positive distances from the target cluster are not checked. This simple strategy has the advantage of being static (since $d_{i,j}$ are compute at index-construction time), but does not guarantee that all the relevant clusters are visited.

Care must be exercised when performing the within-cluster search. In fact, during this step the algorithm computes the distances between the projections of $\mathbf{T}$ and of the cluster points onto a subspace (hyperplane) of the original space. Simple geometry shows that two points that are arbitrarily far apart in the original space can have arbitrarily close projections. In particular, while the points within the cluster are (on the average) close to the subspace by construction, the distance between the query point and the subspace can be large, especially when clusters other than the primary one are searched. The algorithm accounts for this approximation by relying on geometric properties of the space. As shown in Figure 2 for a 2-dimensional table, the distance D_P between $\mathbf{T}$ and a point $\mathbf{P}$ on the subspace (in the figure, the centroid), can be computed using the Pythagorean theorem, as $D_P^2 = D_s^2 + D'^2$, where D_s is the distance between $\mathbf{T}$ and the subspace and D'^2 is the distance between the projection $\tilde{\mathbf{T}}'''$ of $\mathbf{T}$ and $\mathbf{P}$. Thus, during the within-cluster search,the quantities $\sqrt{D'^2 + D_s^2}$ are substituted for D'^2. To compute D_s^2, the Pythagorean theorem is invoked again: the cluster centroids $\overline{\mu}_i$ are known (in the

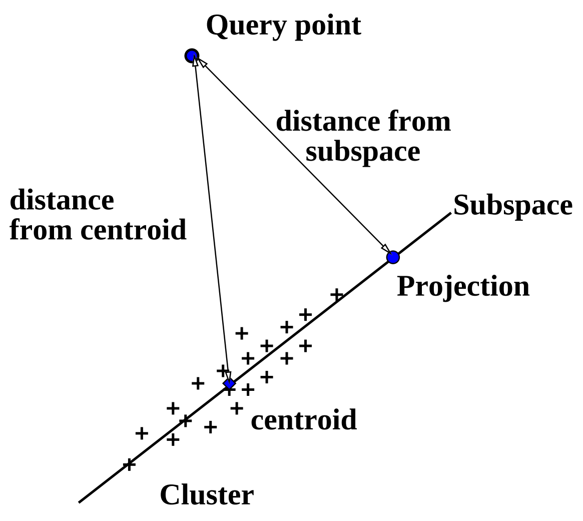

Figure 2: Decomposing a template into its projection onto the subspace of a cluster and its orthogonal component.

original space), thus $D_2(\mathbf{T}, \overline{\mu}_i)$ are easily computed. The rotation of the space corresponding to $\mathbf{V}_i$ does not affect the Euclidean distance between vectors, thus the distance between $\mathbf{T}$ and the subspace is just the distance between $\tilde{\mathbf{T}}''$ and the projection $\mathbf{T}'''$. This quantity can be easily computed during the dimensionality reduction step, by squaring the values of the discarded coordinates, adding them and taking the square root of the result.

4 Performance Study

Before proceeding with the main discussion, we provide some background material about the numerical packages and the input data used in our study.

Although the experiments have been performed on a number of datasets, results from a dataset with 16,129 rows and 60 texture features obtained with a Gabor decomposition [1] are presented here. These features are extracted from a Synthetic Aperture Radar (SAR) image of size 512×512 pixels taken from Alaska.

In what follows, the effect of CSVD on the reduction in the volume and preserved variance is explored first. The effect that the approximate indexing has on the efficiency of retrieval (i.e., the precision for a given recall level) is then investigated.

4.1 The Tradeoff between Volume and Preserved Variance

The relationship between the fraction of total preserved variance (F_{var}) and the number of retained dimensions (p) and equivalently the fraction of retained volume (F_{vol}) follows from the discussion in Section 2.2. There is a need to quantify this relationship, since SVD and hence CSVD would only be desirable if the *fraction of discarded total variance*, denoted by $F_{disc} = 1 - F_{var}$, is small for high data compression ratios.

In Figure 3, F_{vol} versus F_{disc} is plotted as a function of the degree of clustering K (percentages rather than fractions are given here and the graphs that follow). There is a significant drop in F_{vol} for even small values of F_{dist}, suggesting that most of the variance in this dataset is concentrated in a

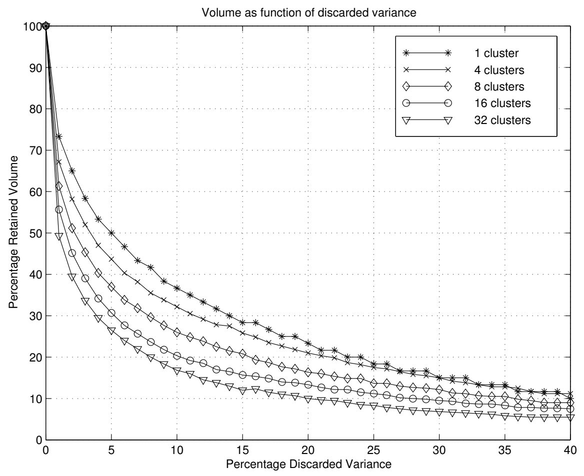

Figure 3: Retained volume as a function of the discarded variance, for different number of clusters.

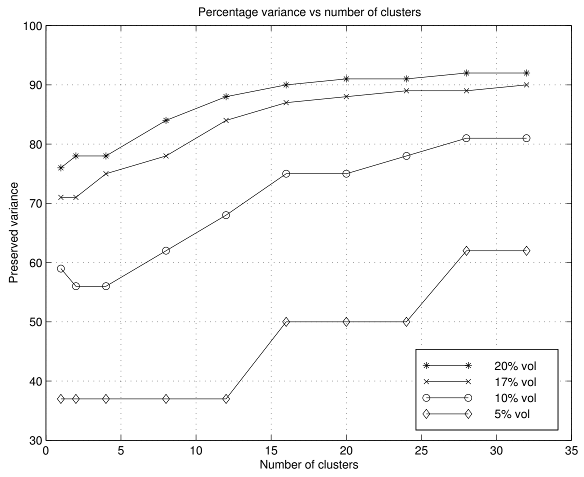

Figure 4: Percentage of retained variance versus number of clusters, for different percentages of retained volume.

few dimensions of the transformed features. The reduction in F_{vol} with F_{disc} is higher for a higher degree of clustering. For example, $F_{dist} = 0.05$ results in $F_{vol} \approx 0.50$ for $K = 1$ cluster, and $F_{vol} \approx 0.27$ for $K = 32$ clusters, indicating that CSVD outperforms SVD to a significant extent. The difference between the two extreme values for F_{vol} (with $K = 1$ and $K = 32$) decreases with the increase of F_{disc}, as can be observed from the figure. But even small difference are significant for very large datasets.

In Figure 4, F_{var} versus the number of clusters (K) is plotted for different volume reduction objectives (F_{vol}). It is observed that CSVD outperforms SVD in the region of interest as there is a substantial increase in F_{var} as K increases for smaller values of F_{vol}. For example, for $F_{vol} = 0.05$, F_{var} increases from $F_{var} = 0.37$ for one cluster to $F_{var} = 0.62$ for $K = 28$ clusters, a 70% increase. A point of diminishing returns is eventually reached with increasing K in all cases, and this happens for smaller K with larger values of F_{vol}. Similar results are obtained with even larger datasets, with over 200,000 rows and 42 dimensions, but the optimum value for K depends on the dataset, F_{vol}, and the quality of the clustering method (see Section 5).

The choice of an optimal K is clearcut in some cases as when $F_{vol} = 0.05$, i.e., $K_{opt} = 28$. Otherwise, tolerating a small decrease in F_{var}, (say within 5% of the maximum) leads to a much smaller value for K, e.g., $K = 16$ with $F_{var} = 0.75$ versus $K = 28$ with $F_{var} = 0.81$ when $F_{vol} = 0.10$. The procedure outlined in Section 3 implements the search for the optimum K.

4.2 The Effect of Approximate Search on Precision

Reducing the dimensionality while improving performance makes the search approximate. More explicitly, the inter-object distances computed using Eq. (1) will be generally different in the primary and secondary vector spaces. These differences are acceptable as long as relative distances with respect to the query point remain the same, i.e., the original ranking of points is preserved. [1] However, larger errors in absolute distances lead to discrepancies in relative distances and erroneous rankings. For example, from the retrieved $k = 20$ NNs, say only $k' = 16$ belong to the "top 20," the *recall* of this retrieval is then $R = k'/k = 0.8$ (see Section 2.1 for definition). A larger number of NNs, say $n = 30$, should be retrieved to attain a pre-specified recall level, say $R \geq 0.9$. The efficiency of the retrieval or *precision*, using the definition in Section 2.1 is then $P = k'/n = 0.6$.

In estimating P, a threshold is specified for the recall ($R_{threshold}$), which is to be met or exceeded. If a search for $N_{init} = k$ points yields $k' \leq k$ correct points such that $R = k'/k \geq R_{threshold}$, then $P = k'/k$. Otherwise, let $N_{final} > N_{init}$ denote the smallest number of retrieved points which yields the desired recall objective. Consequently, $R = k'/k \geq R_{threshold}$, then $P = k'/N_{final}$.

The NN query is repeated with a sufficiently large number (I) query points to obtain an accurate estimate for the mean value of precision ($\overline{P}$). At the i^{th} experiment $N_t(i)$ vectors are retrieved in order to find at least $R_{threshold} \times k$ NNs. Let $N_a(i) \geq kP_{threshold}$ denote the number of NNs obtained in the i^{th} experiment. Then the precision metric for this experiment is $p(i) = N_a(i)/N_t(i), 1 \leq i \leq I$ and the mean precision is given by $\overline{P} = \sum_{i=1}^{I} p(i)/I$.

[1]In fact we are really interested in retrieving the k NNs, rather than rankings, but it is easy to see that erroneous ranking will lead to lower recall, especially for smaller values of k.

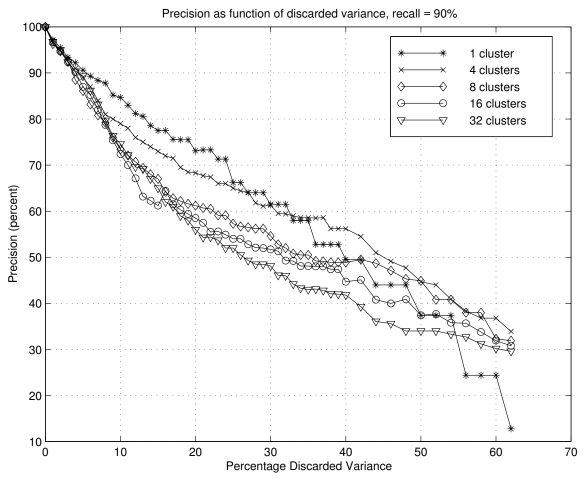

Figure 5: Precision as function of retained variance, for different number of clusters.

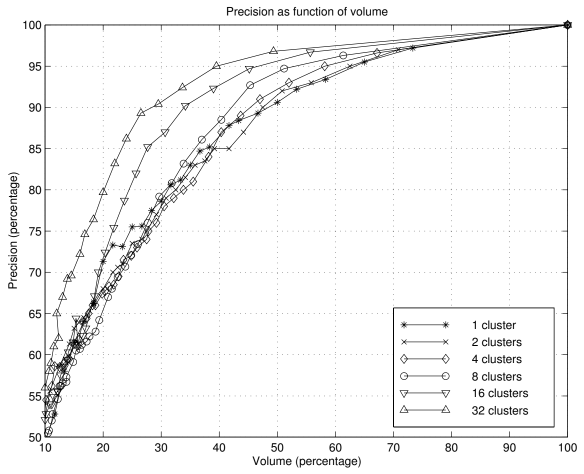

Figure 6: Precision as function of the retained volume, for different number of clusters.

At this point, experimental results that characterize the precision for NN queries with $k = 20$ and $R_{threshold} = 0.9$ are reported. The number of experiments to obtain an accurate estimate of $\overline{R}$ depends on the characteristics of the table. Experiments with the above dataset showed that $I = 100$ yields sufficiently tight confidence intervals for $\overline{R}$ (at a 90% confidence level).

Figure 5 depicts the mean precision ($\overline{P}$) as a function of the fraction of the total discarded variance (F_{disc}) for various number of clusters (K). Firstly, it is observed that $\overline{P} > 0.5$ for F_{disc} up to 0.4, which is very encouraging. A single cluster provides the highest precision when F_{disc} is small. This can be attributed to the inaccuracies in reconciling inter-cluster distances and results in reduced precision (see Figure 2). When V_{disc} is large, clustering improves precision as the NNs of a point are usually in the same cluster.

In Figure 6, the mean precision as a function of F_{vol} is plotted as a function of the number of clusters. It is interesting to note that $K = 32$ clusters result in the highest precision. For the same volume reduction level, the best information is preserved with a higher number of clusters.

This was to be expected, since with an increasing number of clusters a larger fraction of total variance (F_{var}) is preserved, leading to a higher precision. It is important to note that even when the data compression ratio is 10:1 the precision remains greater than 50%.

In an operational system, the number of points to be retrieved to attain $R_{threshold}$ need to be determined for a given query point $\mathbf{T}$, the number of NNs k, and the required recall $R_{threshold}$. Given that $\overline{P}$ is saved for a given $R_{threshold}$, a total of $n = k/\overline{P}$ points can be searched. Alternatively, given that the histogram of precision values is available, we can use the conservative approach of retrieving $n = k/P_{min}$, where P_{min} denotes the minimum precision in a set of experiments.

5 Summary

Many emerging multimedia database and data mining applications require efficient similarity retrieval techniques from very high dimensional datasets. In this paper, a novel methodology, *Clustering Singular Value Decomposition (CSVD)*, is proposed for approximate indexing of numeric tables with high dimensionality. This method achieves additional dimension reduction as compared to those SVD-based methods through exploiting the clustered nature of many heterogeneous datasets. In the proposed method, the feature space is first partitioned into multiple subspace and dimension reduction technique such as SVD is then applied to each subspace.

The effectiveness of the proposed CSVD method is validated with datasets consisting of feature vectors extracted from a benchmark satellite image collections that consist of 37 image clips. With similar precision vs. recall performance, it is shown in this paper that the CSVD method preserves from 37% to 62% more variance than SVD for a 20-fold dimensionality reduction. Conversely, when the reduction in variance is kept at a constant 40%, SVD is shown to require, on the average, twice as many dimensions as CSVD.

Several extensions to this work are currently in progress. Firstly, explore the performance of SVD-friendly clustering methods, i.e., methods that create clusters with an eye on reduced dimensionality. Secondly, investigate the effect of multi-level clustering and SVD on performance and develop procedures for constructing an optimal index in this case. The results from both investigations will be reported in a future paper.

References

[1] Beatty, M., and Manjunath, B. Dimensionality reduction using multi-dimensional scaling for content-based retrieval. In *Proc. IEEE Int'l Conf. Image Process. ICIP'97* (Santa Barbara, CA, Oct. 1997), vol. 2, pp. 835–838.

[2] Berchtold, S., Keim, D. A., and Kriegel, H.-P. The X-tree: An index structure for high-dimensional data. In *Proc. 22nd Int'l Conf. on VLDB* (Bombay, India, Sept. 1996), pp. 28–39.

[3] Berchtold, S. et al. A cost model for nearest neighbor search in high-dimensional data space. In *Proc. 16th ACM Symp. on Principles of Database Systems* (Tucson, AZ, May 1997), pp. 78–86.

[4] Berchtold, S. et al. Fast parallel similarity search in multimedia databases. In *Proc. SIGMOD Conf. 97* (Tucson, AZ, May 12-15 1997), pp. 1–12.

[5] Blott, S., and Weber, R. A simple vector-approximation file for similarity search in high-dimensional vector spaces. Tech. rep., Institute of Information Systems, ETH, Zurich, Switzerland, 1997.

[6] Cherkassky, V., Friedman, J., and Wechsler, H. *From Statistics to Neural Networks: Theory and Pattern Recognition Application.* Springer-Verlag, New York, 1993.

[7] Everitt, B. *Cluster Analysis*, 3rd ed. John Wiley & Sons, 1993.

[8] Faloutsos, C. *Searching Multimedia Databases by Content.* Kluwer Academic Publishers, Boston, 1996.

[9] Gaede, B. V., and Gunther, O. Mutidimensional access methods. *ACM Computing Surveys* (to appear in 1998).

[10] Jolliffe, I. T. *Principal Component Analysis.* Springer-Verlag, New York, 1986.

[11] Kim, B. S., and Park, S. B. A fast k nearest neighbor finding algorithm based on the ordered partition. *IEEE Trans. on Pattern Analysis and Machine Intelligence PAMI 8*, 6 (Nov 1986), 761–766.

[12] Korn, F., Jagadish, H. V., and Faloutsos, C. Efficiently supporting ad hoc queries in large datasets of time sequences. In *Proc. ACM SIGMOD Int'l Conf.* (Tucson, AZ, May 1997), pp. 289–300.

[13] Linde, Y., Buzo, A., and Gray, R. An algorithm for vector quantizer design. *IEEE Trans. Communications COM-28*, 1 (Jan. 1980), 84–95.

[14] Press, W.H. et al. *Numerical Recipes in C.* Prentice-Hall, 1993.

[15] Russopoulos, N., Kelley, S., and Vincent, F. Narest neighbor queries. In *Proc. ACM SIGMOD Int'l Conf.* (San Jose, CA, May 1995), pp. 71–79.