Near-Optimal Partial Linear Scan for Nearest Neighbor Search in High-Dimensional Space

Jiangtao Cui[1], Zi Huang[2], Bo Wang[1], and Yingfan Liu[1]

[1] School of Computer Science and Technology, Xidian University
Xi'an 710071, China
{cuijt,wangbo_st,yfliu_st}@xidian.edu.cn
[2] School of Information Technology and Electrical Engineering,
The University of Queensland, QLD 4072, Australia
huang@itee.uq.edu.au

Abstract. One-dimensional mapping has been playing an important role for nearest neighbor search in high-dimensional space. Two typical kinds of one-dimensional mapping method, direct projection and distance computation regarding to reference points, are discussed in this paper. An optimal combination of one-dimensional mappings is achieved for the best search performance. Furthermore, we propose a near-optimal partial linear scan algorithm by considering several one-dimensional mapping values. During the linear scan, the partial distance to the query point computed in the 1D space is used as the lower bound to filter the unqualified data points. A new indexing structure based on clustering with Gaussian Mixture is also designed to facilitate the partial linear scan, which can reduce both the I/O cost and distance computations dramatically. Comprehensive experiments are conducted on several real-life datasets with different dimensions. The experimental results show that the proposed indexing structure outperforms the existing well-known high-dimensional indexing methods.

Keywords: indexing methods, nearest neighbor search, one-dimensional mapping.

1 Introduction

k-Nearest Neighbor (k-NN) search in high-dimensional space has many applications such as multimedia retrieval, time-series matching, data mining, and the like. One serious problem in achieving efficient k-NN search is the notorious "curse of dimensionality". The traditional hierarchical indexing structures always degenerate to visiting the entire dataset when the dimensionality is high, and are eventually outperformed by linear scan[5]. Therefore, for k-NN queries in high-dimensional space, linear scan method remains an efficient search strategy for similarity search [4] [5].

The design of indexing algorithm is also governed by hardware constraints. For the disk-based indexing structure, I/O operations often dominate the cost

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 101–115, 2013.
© Springer-Verlag Berlin Heidelberg 2013

of query processing because of the relatively low transfer speed between memory and disk. The linear scan avoids seeking the specified page, and it is much faster than random access. Furthermore, linear scan methods are a lot easier to integrate within a database engine or query engine. The typical linear scan methods are the VA-file approachs [20] [8] and the omni-sequential. However, these methods usually need to linearly scan the whole dataset, and every point in the dataset incurs expensive distance computations.

Recently, one-dimensional mapping methods have attracted the attention, which are widely used for the exact or approximate k-NN search. Two typical one-dimensional mapping methods in metric space are projection-based techniques and distance-based techniques. For the approximate NN search, both the Locality Sensitive Hashing (LSH) [10] and NV-tree [14] are based on the concept of projecting data points onto a line and classifying locations along this line with different symbols. The random projection is applied in LSH methods, and Principal Component Analysis (PCA) is used in NV-tree. For the exact NN search, one-dimensional mapping is a good choice for the filter-and-refine strategy, because the 1D mapping values have the lower-bounding property. When mapping the high-dimensional data to 1D space, only partial data points need to be accessed during the query. The omni-sequential [9] and iDistance [12] are two typical methods using distance-based mapping.

When sorting the 1D mapping values, a simple linear scan algorithm using filter-and-refine model can be presented. During the search, we locate the first accessed point with its 1D value nearest to the mapping value of the query, and then perform a two-stage linear scan. Since the 1D mapping value has the lower-bounding property, only partial data file need to be linearly scanned. We call this search strategy as "Partial Linear Scan (PLS)". The PLS has the advantage of linear scan, and it also avoids accessing the whole data file.

In this paper, we aim at finding a best mapping scheme to support PLS. Two kinds of mapping methods were analyzed and compared in this paper. To the best of our knowledge, the performance of different one-dimensional mappings is firstly studied in this paper, and our observations will be helpful for the researches and applications in the high-dimensional space. To summarize, we make the following main contributions:

- We formalize the PLS algorithm with respect to one-dimensional mapping and discuss its advantages over existing linear scan approaches.
- We identify the parameter that affect the performance of PLS and present a near-optimal PLS to take both processor and I/O time into account. By using the variance of 1D mapping values as the parameter, we find the optimal reference point for distance-based mapping and the optimal projection line for projection-based mapping. Furthermore, an optimal combination of different 1D mapping values is observed.
- We present PLS-VA, an efficient indexing structure for high-dimensional datasets. Clustering with Gaussian Mixture is applied in PLS-VA and experiment results on multimedia datasets show it can facilitate the PLS on the approximate vectors.

The rest of this paper is organized as follows. In Section 2, we survey the main current indexing methods. Some observations of our work and the PLS algorithm are presented in Section 3. Section 4 introduces our PLS-VA indexing structure. An extensive performance study is reported in Section 5, and finally, we conclude our paper in Section 6.

2 Related Work

Research on high-dimensional indexing can be divided into exact and approximate retrieval. In the approximate category, Hashing has been demonstrated to be effective for similarity search. LSH is one typical hashing method [10]. In Euclidean space, the basic idea is to project data points onto a line and classify locations along this line with different symbols. Recently, the distance-based hashing has also been introduced [1]. Several heuristic variants of LSH have also been suggested. For example, Multi-probe LSH can obtain the same search quality with much less tables [15], while LSB-tree addresses both the quality and efficiency of multimedia retrieval [19]. NV-tree is another representative index for approximate NN search [14]. Using partitioning and projections based on PCA, NV-tree can give approximate answers with a single random disk read.

For exact k-NN queries in high-dimensional space, there exist mainly three categories of high-dimensional indexing methods, such as dimensionality reduction, data approximation and one-dimensional mapping. A well known approach to improving the indexing performance is Dimensionality Reduction (DR) before indexing the data in the reduced-dimensionality space [16]. The linear DR approach first condenses most of information in a dataset to a few dimensions by applying PCA or other techniques. Two strategies for dimensionality reduction include Global DR and Local DR [7].

VA-file is the representative high-dimensional index for data approximation, which suggests accelerating the linear scan by the use of data compression and filtering of the feature vectors [20]. Some extensions of VA-file have been proposed, such as IQ-tree [3], which achieves better query performance by combining a tree structure with VA-file. VA$^+$-file improves the approximate ability of VA-file by transforming the data points in PCA space [8]. The Vector Approximation can be seen as the scalar quantization. The Hyperplane Bound (HB) method uses vector quantization to compress data points, and a new filtering algorithm based on bounding hyperplane has been presented [17].

One-dimensional mapping approaches provide another direction for high-dimensional indexing. The 1D mapping value has the lower-bounding property. Therefore, data points can be cut off based on the 1D values, and the real nearest neighbors are verified in the set of candidates. The typical example is iDistance [12]. The dataset is partitioned and a reference point of each partition is defined. Then data points are mapped to 1D values based on their distance to the reference point. However, its performance is sensitive to the selection of reference points and too much random access of data pages is required. Omni-sequential method chooses some reference points as global 'foci' and gauges all

other data points based on their distances to each focus [9]. During the search, the distances of the query point to each focus are computed, and the triangular inequality can be used to reject the impossible point.

3 Optimal One-Dimensional Mapping

3.1 One-Dimensional Data Transformation

The basic idea of one-dimensional mapping is transforming high-dimensional points into 1D values, which can be used to compute the lower bound of the distance between points in the original high-dimensional space. Two typical one-dimensional mapping methods in metric spaces are projection-based techniques and distance-based techniques with respect to a chosen reference point.

Definition 1 (Projection-based 1D Transformation). *Given a point p and a vector X in the high dimensional space $\boldsymbol{R}^d$, p can be projected onto X with its one-dimensional mapping value $p \cdot X$.*

Definition 2 (Distance-based 1D Transformation). *Given a point p and a reference point o in the high dimensional space $\boldsymbol{R}^d$, the one-dimensional mapping value of p is defined as the distance between p and o, which is denoted as $dist(p, o)$. For Euclidean distance, $dist(p, o) = ||p - o||$, where $|| \cdot ||$ is L_2 norm.*

Lemma 1. *Given two points p and $q \in \boldsymbol{R}^d$, whose 1D mapping values are denoted as p^1 and q^1 respectively, we have $dist(p, q) \geq |p^1 - q^1|$*

Proof. There are two cases need to be considered here.
(1) For projection-based transformation, $p^1 = p \cdot X$ and $q^1 = q \cdot X$. We have

$$dist(p, q) = ||p - q|| \geq ||pX - qX|| = |p^1 - q^1|$$

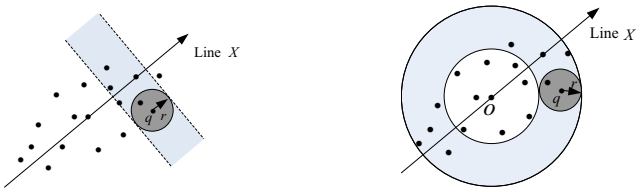
(2) For distance-based transformation, $p^1 = dist(p, o)$ and $q^1 = dist(q, o)$. By considering the triangular inequality, we have

$$dist(p, q) \geq |dist(p, o) - dist(q, o)| = |p^1 - q^1|$$

3.2 Partial Linear Scan

As proved in the Lemma 1, given two points, their full distance in the original high dimensional space cannot be smaller than their 1D transformation distance. Thus, their 1D transformation values (or 1D values, in short) can be used to compute the lower bound of the full distance. According to this property, a synchronous bi-directional linear scan algorithm can be designed to perform an efficient *PLS*.

Given a collection of data points, they are firstly sorted in ascending order according to their 1D values. Given a query point q, a distance array $kDist[\,]$ of size k is employed to store the k-NN distances found so far. During the search,



(a) Projection (b) Distance

Fig. 1. Search Space after 1D transformation

we locate the first point to be accessed as p_b whose 1D value p_b^1 is nearest to q^1. A bidirectional scan will be conducted by accessing data points from p_b (i.e., forward and backward from p_b). In practice, the linear scan in the disk always accesses data pages along one direction. The bidirectional search can be modified as two search processes with the same direction. The detailed algorithm is shown in Algorithm 1 and explained below.

Require: Query point q, dataset D
Ensure: $kDist\,[]$
1: Initialize $kDist\,[]$ with $MAXREAL$
2: Locate the first accessed point p_b w.r.t q^1 in the first stage
3: **for** $i = b$ to N **do**
4: Calculate $\left|p_i^1 - q^1\right|$
5: **if** $\left|p_i^1 - q^1\right| > kDist[k]$ **then**
6: break //End of the first stage scan
7: **else**
8: Calculate $dist\,(q, p_i)$ and update $kDist\,[]$
9: **end if**
10: **end for**
11: Locate the first accessed point p_s w.r.t q^1 and $kDist[k]$ in the second stage
12: **for** $i = s$ to $b - 1$ **do**
13: Calculate $\left|p_i^1 - q^1\right|$
14: **if** $\left|p_i^1 - q^1\right| > kDist[k]$ **then**
15: break //End of the second stage scan
16: **else**
17: Calculate $dist\,(q, p_i)$ and update $kDist\,[]$
18: **end if**
19: **end for**

Algorithm 1. Partial linear scan for exact kNN search

We perform the first stage scan first (Line 3-10). Before computing the full distance between the query q and the current data point p_i, the distance between q^1 and p_i^1 is calculated firstly (Line 4). If $\left|p_i^1 - q^1\right| > kDist\,[k]$, p_i can be safely pruned without computing its full distance $dist\,(q, p_i)$ of all dimensions. Furthermore, the linear scan in this direction can be terminated (Line 5-6). If $\left|p_i^1 - q^1\right| < kDist\,[k]$, the full distance between p_i and q is required to be calculated while $kDist\,[\,]$ will be updated (Line 8). When the scan in the first stage

is terminated, we perform the second stage scan similarly (Line 11-18). The first accessed point p_s at the second stage satisfies $\left|p^1_{s-1} - q^1\right| > kDist[k]$ and $\left|p^1_s - q^1\right| \leq kDist[k]$. Fig. 1 shows data points accessed during the PLS when performing projection-based and distance-based 1D transformation respectively, where the k-th NN distance is denoted as r.

3.3 Performance Analysis on One-Dimensional Mappings

In this subsection, we will discuss how to evaluate the performance of different 1D transformation methods and introduce several important observations. The filtering efficiency is a critical indicator of the PLS performance based on a certain 1D data transformation. It is formally defined as below.

Definition 3 (Filtering Efficiency, FE). *Given a number of N data points and their 1D transformation, N_f is the number of data points pruned in PLS. The filtering efficiency (fe) is defined as: $\boldsymbol{fe} = N_f/N$, where $N = |D|$.*

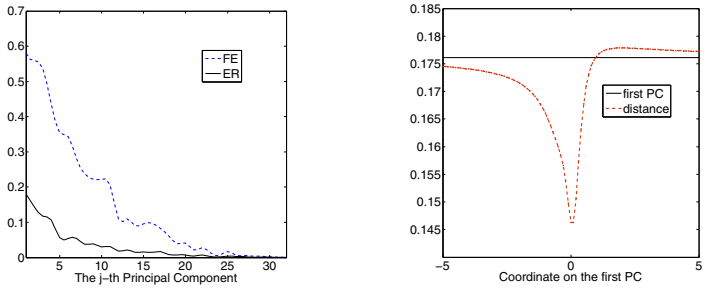
The best and most widely used approach for data projection is based on PCA. For any two points p and q in a dataset $D \in \mathbf{R}^d$, they can be projected onto the j-th PC of D, denoted as p_j and q_j respectively. We give another difinition according to the PCA.

Definition 4 (Energy Ratio, ER). *Let σ_j^2 denotes the variance along the j-th PC. The energy ratio of the j-th PC, denoted as $er\,(j)$, is defined as follows: $er\,(j) = \sigma_j^2 / \sum_{k=1}^{d} \sigma_k^2$*

Apparently, a larger fe implies a smaller number of data points to be accessed during the PLS. The $er(j)$ measures the percentage of the variance introduced by j-th PC over the whole distance variance. It's difficult to analyze the **FE** theoretically, since datasets have different distributions and different query points have different query performance. However, several important observations have been found according to the experiment results. The first group of experiments are conducted on a widely used real-life dataset, COLHIST[1], which contains a number of 68,040 32-dimensional color histograms. The conclusions and observations found on the COLHIST dataset are also validated by the experiments on other datasets such as, the LANDSAT dataset and the SIFT dataset. To get fairly results, 500 10-NN searches are performed to get the average performance. We first test the **FE** and **ER** along different PCs, which are shown in Fig. 2(a). We can observe that two curves have similar tendency and a larger **ER** leads to a higher **FE**.

Observation 1. *For projection-based 1D data transformation, the ER can reflect the FE of the PCs. In result, the first PC is the optimal projection line for 1D data transformation.*

[1] `http://kdd.ics.uci.edu/database/CorelFeatures`



(a) FE and ER (b) Variances comparison

Fig. 2. Performance Analysis

The Observation 1 can be easily justified. Data variance measures how data points are spread out along the corresponding direction. As shown in Fig. 1(a), the shadow area is the search space of the query q. The larger the data variance is along X, the fewer the data points fall into the shadow area, leading to a higher **FE**.

For distance-based 1D transformation, the **FE** mainly depends on the choice of reference point. We have shown the strong correlation between the data variance and the filtering ability. An optimal reference point should be the point maximizing the variance of the distances from the data points to the reference point. It is impossible to test the entire space to find the optimal reference point. Fortunately, PCA can be used to find the direction with largest data variance for a dataset. The optimal reference point most likely lie on the line identified by the first PC [18]. We select points along the first PC as the reference points, and test the corresponding data variances. Fig. 2(b) shows the variances of the distances w.r.t a reference point when selecting the reference point along the first PC. The tests on the the other PCs also show the similar tendency.

Observation 2. *The largest variance of distances w.r.t the reference point lies out of one side of the line determined by the first PC. When the reference point lies very faraway the origin along the first PC, the limit of variance of distances is equal to the variance of the first PC.*

3.4 Combination of Several One-Dimensional Mappings

In this subsection, we will focus on 1D data transformation by utilizing both project-based and distance-based mapping to achieve a better filtering efficiency. By taking more information into account, the data points can be better distinguished. Two models are studied in our work. In the first model, we consider both 1D values computed by data projection and the distance to a reference point. In the second one, we select two reference points to get distances. The examples of these two advanced 1D transformation models are shown in Fig. 3, in which only data points labeled with asterisk need to be accessed. We firstly

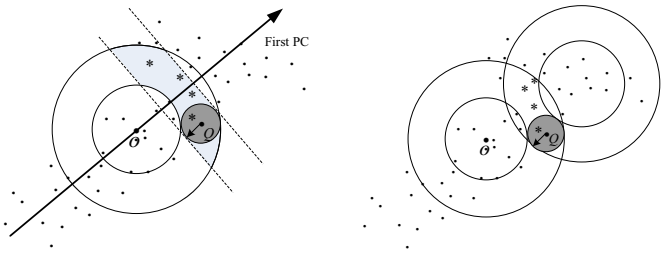


Fig. 3. Combination of 1D transformation models

test the combined **FE** of projection and distance by selecting reference point along the first PC. Fig. 4(a) shows the results.

Observation 3. *When combining 1D values achieved by projection-based and distance-based 1D transformations, the largest **FE** can be reached when the reference point lies on the origin and the projection line is along the first PC.*

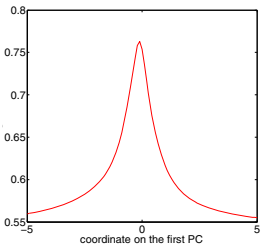
The distance of data points to the origin is the norm. When the reference point lies far outside of origin, the accessed region using projection and distances will overlap, and result in almost the same **FE** as that just using projection.

Now, we will discuss how to get the largest **FE** when selecting two reference points. By fixing one reference point on the first PC, we test the combined **FE** when selecting the other reference point along the first PC. Three typical points are chosen as the first reference point to illustrate different curve tendency, whose 1D value on the first PC are 0, 0.5 and 3 respectively. Fig. 4(b) shows the **FE** when one reference point is fixed.

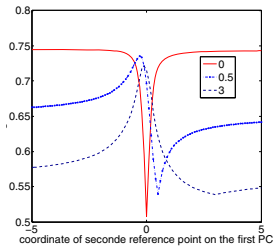
When the first reference point lies in the origin (i.e., 0), the largest **FE** can be achieved if the second reference point locates very far outside of the origin. As mentioned above, when the reference point lies very far outside of origin, the accessed region using distances and projection will fully overlap. Therefore, we can get the last observation.

Observation 4. *When 1D transformation is based on the distance w.r.t two reference points, its highest **FE** cannot be greater than the highest **FE** achieved by utilizing both projection-based and distance-based 1D transformations.*

So far, we can design a near-optimal PLS algorithm to take both processor and I/O time into accout. Since combination of projection on the first PC and norm can get better **FE**, and the variance of first PC is larger than the norm as shown in Fig. 2(b), we sort data points according to the projections on the first PC, and the norms of data points are also embedded. This algorithm is not optimal if just considering I/O cost, since the largest variance of distances is not applied. When performing k-NN search, the projection values on the first PC are used for terminating the searching process. And the norms are used to reject more impossible candidate points. During the high-dimensional distance computation, the Partial Distortion Search (*PDS*) algorithm can be adopted which use the



(a) Combining projection and distance



(b) Combining two distances

Fig. 4. Filtering efficiency

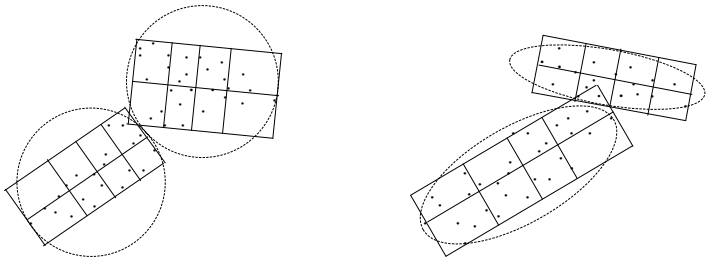
projections on the other PCs to reject points. The *PDS* algorithm was widely used in the Vector Quantization encoding process [2]. Since most of energy is condensed to the first several PCs, we can perform *PDS* algorithm only on the first several PCs. If data points can not be rejected on these dimensions, we directly calculate $dist\,(Q, P_i)$ on all dimensions.

4 PLS-VA

We have discussed how to get the near-optimal query performance when performing *PLS*. In this section, we aim to build a new indexing structure applying the PLS. One of the state of the art indices for linear scan is the VA-file approach. Our new indexing structure is proposed by utilizing VA-file and PLS, which is so called PLS-VA.

4.1 The Framework of PLS-VA

The proposed indexing structure is specially designed for real-life datasets, in which PCA is employed to get the projection values on the first PC. To facilitate the PLS, we aim at getting the better **FE** by using partition technique. The similar idea has been presented in iDistance, in which all data points in different partitions are represented in a single dimensional space. It is also well known that LDR can outperform GDR since it can find the correlations in the intrinsic clusters. Several clustering techniques have been investigated, such as K-means clustering and clustering with Gaussian Mixture [6]. We find that the clustering with Gaussian Mixture is more suitable for PLS than K-means clustering. Fig. 5 shows an example. Gaussian Mixture is a model-based clustering approach, which consists in using certain models for clusters and attempting to optimize the fit between the data and the model. The Gaussian mixture architecture estimates probalility density fuctions for each class, and then performs classification based on Bayesian rule.



(a) K-means partition (b) GMM partition

Fig. 5. Different partitions for building approximate vectors

Obviously, building approximate vectors in several partitions separately can also improve the approximate ability, and the additional cost is that several codebooks need to be maintained. After performing the clustering, PCA is applied to each cluster. The approximate vectors belonging to the same cluster are firstly sorted according to the 1D projection values and then loaded in contiguous data pages. The bound of projection values of each page can be described using a two-dimensional array $[\alpha, \beta]$. A B^+-tree is applied to manage all the arrays in one cluster, where only two values in one page are indexed as the keys in the B^+-tree.

4.2 Searching in PLS-VA

When performing k-NN search in the PLS-VA, the accessing order of each cluster is determined by computing the lower bounds between the query point and each cluster. Each cluster can be denoted as a hyper rectangle or a hyper sphere. The hyper rectangle can be seen as an MBR (minimum bounding rectangle) used in the R-tree [11], where the lower bound of distance between q to the MBR can be easily calculated. The lower bound between q and the sphere can also be computed, and the maximum value between two lower bounds is chosen as the real lower bound between q and the cluster.

For the k-NN query in the whole dataset, we use a priority queue to navigate the clusters accessed in increasing order of their lower bounds to the query point. If the lower bound of a cluster to the query point is zero, the distance between the centroid and q can be compared. During the search, if the lower bound of one cluster to q is larger than the k-th smallest distance found so far, all the data points in this cluster do not need to be accessed and the search can be terminated. The k-NN search in PLS-VA includes two phases. The first phase is to linearly scan the partial approximation file, and the second phase is the access of exact vectors. The first search phase can guarantee no false dismissals for the query. However, the search results may contain false positives which have to be further refined in the second phase. In the first stage, it is noted that we need to compute lower and upper bounds of distance instead of exact distances. Therefore, the **FE** of PLS-VA is lower than the PLS in the exact vectors.

5 Experiments

In this section, we performed extensive experiments on the high-dimensional real-life datasets to demonstrate the practical effectiveness of PLS-VA. All experiments were executed on HP workstation xw9300 with AMD Opteron 2.2 GHz CPU, 2GB RAM and 73GB 10Krpm SCSI disk. The data page size used in the experiments is 4096 Bytes. We use real-life datasets to evaluate the effectiveness of our method. Two datasets which are widely used in high-dimensional indexing were chosen. The first dataset is **COLHIST**[2], which contains 68,040 32-dimensional color vectors. The other dataset is called Satellite Image Texture (**LANDSAT**)[3], which contains 275,245 60-dimensional feature vectors extracted from satellite images.

We use 500 queries to obtain the average results on 10-NN, 20-NN and 50-NN search. A comprehensive performance study has been conducted on VA, iDistance, HB, and PLS-VA. The results show the superiority of PLS-VA.

5.1 I/O Cost Model

The search in PLS-VA is comprised two phases. The first phase is to linearly scan the approximate file, and the second phase is the random access of exact vectors. We use the random page access as the I/O metric. The total I/O cost of PLS can be calculated as: $IO_{total} = IO_l + IO_r$. Suppose there are M clusters, we need compute the sum of linear I/O cost of M clusters. Accessing of a new cluster also need additional 2 random page access. The total I/O cost of PLS-VA can be calculated as: $IO_{total} = IO_l + IO_r + 2 \times M$. Assuming linear I/O is 10 times faster than random I/O [7] [13], and the post processing of candidate can be one I/O per exact vectors [20]. However, as observed in [7], this assumption is overly pessimistic. We, therefore, can assume that the total I/O cost of PLS-VA is: $IO_{total} = num\_l/10 + num\_r/2 + 2 \times M$. Where num_l is the total number of page accessed in the first phase, and num_r is the number of exact vector accessed during the second searching process.

5.2 Performance Study on the Filtering Efficiency

In the first experiment, we study the effect of different clustering methods. The results of 10-NN queries of PLS algorithm are shown in Fig. 6. We can see that the clutering with GMM can improve the **FE** compared with the K-means clustering. As expected, as the number of cluster increases, PLS incurs larger **FE**. While we can choose a large number of clusters to improve the **FE**, this will increase the random I/O cost. So a moderate number of clusters is fine. In our other experiments, we have used 4 as the default number of clusters.

The second experiment studies the **FE** of different methods. We compare PLS with two other well known indexing methods using clustering technique,

[2] `http://kdd.ics.uci.edu/database/CorelFeatures`
[3] `http://vision.ece.ucsb.edu/datasets/`

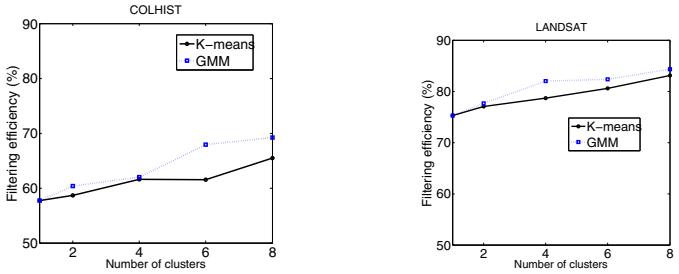


Fig. 6. Comparison of filtering efficiency, the result of 10-NN queries

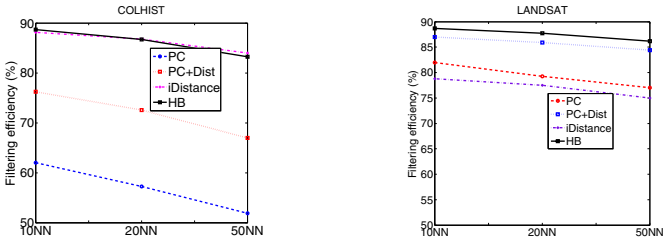


Fig. 7. Comparison of filtering efficiency

iDistance and HB, which will be discussed later in detail. Both iDistance and HB use the filter-and-refine strategy, and need to select a good number of clusters. We use 64 partitions for iDistance in two datasets. We also tried the HB for several numbers of clusters, then use 80 clusters for the COLHIST and 200 clusters for the LANDSAT. We also studied the effect of combination of two one-dimensional mappings. The results are shown in Fig. 7. The "PC" denotes the **FE** of the first PC, and "PC+Dist" deontes the **FE** when combining the first PC and the norm. The combination of several 1D mapping values can improve the **FE**. From the figures, we can see very small difference between "PC+Dist" and HB, and HB performs the best for the two datasets.

5.3 Comparative Study of PLS-VA with Other Methods

In this section, we compare PLS-VA with VA-file, iDistance and HB. For VA and PLS-VA, the average approximate bit length is 6 bits per dimension for the two datasets. When using PLS algorithm in PLS-VA, the distortion distance on the first 5 PCs are used to prune data points. iDistance need to select a good number of reference points to work efficiently. In our comparative studies, we choose 64 reference points, which was reported to have the best average performance. In COLHIST, the first search radius and the increasing step are 0.01, and in LANDSAT the same parameters are set 0.1. HB uses the separating hyperplane boundaries of Voronoi clusters to complement the clustering-based index. However, the HB suffers a huge number of lower bounds computations, suppose

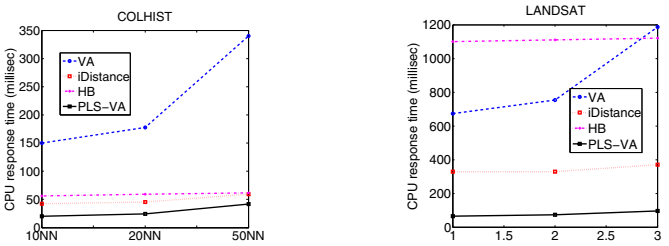


Fig. 8. Comparison of CPU cost

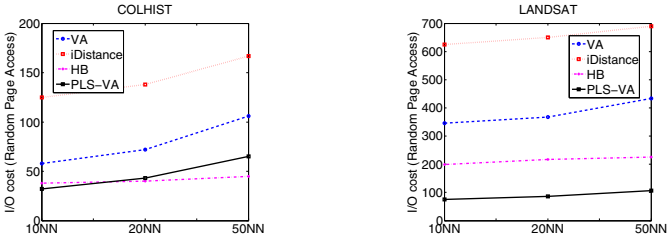


Fig. 9. Comparison of I/O cost

there are K clusters, and as many as $K(K-1)/2$ lower bounds calculations are needed during the query. As mentioned above, 80 clusters are applied in COLHIST and 200 clusters in LANDSAT.

First we present the comparision of CPU response time of different methods. The results are shown in Fig. 8. The **FE** of HB and iDistance is larger than PLS-VA, but PLS-VA has the shortest response time. It has a speedup factor of more than 2 over iDistance and 6 over VA. The CPU response time of HB is relative to the number of clusters. HB does not yet provide effective pruning mechanisms of data points in the candidate cluster. To achieve better **FE**, more clusters are needed. It can be seen its CPU time is more than 1 second in the LANDSAT. PLS-VA is better than iDistance because the combination of several 1D mapping values enables more effective pruning of data points.

We also compared the I/O cost between different methods. The results are shown in Fig. 9. Both PLS-VA and HB have less random page accesses than VA and iDistance. The iDistance has the most random page accesses, because iDistance incrementally enlarges the search space to find NNs in different partitions, and more partitions result in more fragmented pages and lead to more random page accesses. HB linearly scan the candidate clusters to find NNs, therefore it has less I/O cost. PLS-VA and HB have similar I/O cost in COLHIST, and PLS-VA has the least I/O cost in LANDSAT. To summarize, PLS-VA outperforms VA and iDistance in both CPU and I/O cost, and PLS-VA has less total response time than HB.

6 Conclusions

In this paper, we have studied the performance of different 1D transformation methods and derived a novel yet more effective model to transform high dimensional data points into 1D subspace, in which an efficient PLS can be performed. A novel indexing structure PLS-VA is proposed which partitions the original data space with GMM and builds approximate vectors on each cluster separately. The proposed indexing structure can be easily built, and it highly improves the pruning power of the linear scan on the vector approximate file. Extensive experiments are conducted on two real-life multimedia datasets. The results confirm that PLS-VA outperforms existing indexing structures.

References

1. Athitsos, V., Potamias, M., Papapetrou, P., Kollios, G.: Nearest neighbor retrieval using distance-based hashing. In: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering, pp. 327–336. IEEE Computer Society, Washington, DC (2008)
2. Bei, C.-D., Gray, R.M.: An improvement of the minimum distortion encoding algorithm for vector quantization. IEEE Trans. Communication 33(10), 1132–1133 (1985)
3. Berchtold, S., Bohm, C., Jagadish, H.V., Kriegel, H.-P., Sander, J.: Independent quantization: An index compression technique for high-dimensional data spaces. In: ICDE 2000: Proceedings of the 16th International Conference on Data Engineering, p. 577. IEEE Computer Society, Washington, DC (2000)
4. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is "nearest neighbor" meaningful? In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 217–235. Springer, Heidelberg (1998)
5. Böhm, C., Berchtold, S., Keim, D.A.: Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. ACM Comput. Surv. 33(3), 322–373 (2001)
6. Celeux, G., Govaert, G.: A classification em algorithm for clustering and two stochastic versions. Comput. Stat. Data Anal. 14(3), 315–332 (1992)
7. Chakrabarti, K., Mehrotra, S.: Local dimensionality reduction: A new approach to indexing high dimensional spaces. In: VLDB 2000: Proceedings of the 26th International Conference on Very Large Data Bases, pp. 89–100. Morgan Kaufmann Publishers Inc., San Francisco (2000)
8. Ferhatosmanoglu, H., Tuncel, E., Agrawal, D., Abbadi, A.E.: High-dimensional nearest neighbor searching. Information Systems 31(6), 512–540 (2006)
9. Filho, R.F.S., Traina, A.J.M., Traina Jr., C., Faloutsos, C.: Similarity search without tears: The omni family of all-purpose access methods. In: Proceedings of the 17th International Conference on Data Engineering, pp. 623–630. IEEE Computer Society, Washington, DC (2001)
10. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: Proceedings of the 25th International Conference on Very Large Data Bases, VLDB 1999, pp. 518–529. Morgan Kaufmann Publishers Inc., San Francisco (1999)
11. Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: SIGMOD 1984: Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, pp. 47–57. ACM, New York (1984)

12. Jagadish, H.V., Ooi, B.C., Tan, K.-L., Yu, C., Zhang, R.: iDistance: An adaptive b+-tree based indexing method for nearest neighbor search. ACM Trans. Database Syst. 30(2), 364–397 (2005)
13. Koudas, N., Ooi, B.C., Shen, H.T., Tung, A.K.H.: Ldc: Enabling search by partial distance in a hyper-dimensional space. In: ICDE 2004: Proceedings of the 20th International Conference on Data Engineering, p. 6. IEEE Computer Society, Washington, DC (2004)
14. Lejsek, H., Ásmundsson, F.H., Jónsson, B.P., Amsaleg, L.: Nv-tree: An efficient disk-based index for approximate search in very large high-dimensional collections. IEEE Trans. Pattern Anal. Mach. Intell. 31, 869–883 (2009)
15. Lv, Q., Josephson, W., Wang, Z., Charikar, M., Li, K.: Multi-probe lsh: efficient indexing for high-dimensional similarity search. In: Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB 2007, pp. 950–961. VLDB Endowment (2007)
16. Postma, E.: Dimensionality reduction: A comparative review 10(February), 35 (October 2009)
17. Ramaswamy, S., Rose, K.: Adaptive cluster distance bounding for high-dimensional indexing. IEEE Trans. on Knowl. and Data Eng. 23(6), 815–830 (2011)
18. Shen, H.T., Ooi, B.C., Huang, Z., Zhou, X.: Towards effective indexing for very large video sequence database. In: SIGMOD 2005: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, pp. 730–741. ACM, New York (2005)
19. Tao, Y., Yi, K., Sheng, C., Kalnis, P.: Quality and efficiency in high dimensional nearest neighbor search. In: Proceedings of the 35th SIGMOD International Conference on Management of Data, SIGMOD 2009, pp. 563–576. ACM, New York (2009)
20. Weber, R., Schek, H.-J., Blott, S.: A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: VLDB 1998: Proceedings of the 24rd International Conference on Very Large Data Bases, pp. 194–205. Morgan Kaufmann Publishers Inc., San Francisco (1998)