Journal of Zhejiang University SCIENCE A
ISSN 1673-565X (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Indexing the bit-code and distance for fast KNN search in high-dimensional spaces*

LIANG Jun-jie†[1,2], FENG Yu-cai[1]

([1]*College of Computer Science & Technology, Huazhong University of Science and Technology, Wuhan 430074, China*)
([2]*Faculty of Mathematics & Computer Science, Hubei University, Wuhan 430062, China*)
†E-mail: ljjhubu@163.com

Received Aug. 22, 2006; revision accepted Dec. 5, 2006

Abstract: Various index structures have recently been proposed to facilitate high-dimensional KNN queries, among which the techniques of approximate vector presentation and one-dimensional (1D) transformation can break the curse of dimensionality. Based on the two techniques above, a novel high-dimensional index is proposed, called Bit-code and Distance based index (BD). BD is based on a special partitioning strategy which is optimized for high-dimensional data. By the definitions of bit code and transformation function, a high-dimensional vector can be first approximately represented and then transformed into a 1D vector, the key managed by a B^+-tree. A new KNN search algorithm is also proposed that exploits the bit code and distance to prune the search space more effectively. Results of extensive experiments using both synthetic and real data demonstrated that BD outperforms the existing index structures for KNN search in high-dimensional spaces.

Key words: High-dimensional spaces, KNN search, Bit-code and distance based index (BD), Approximate vector
doi:10.1631/jzus.2007.A0857 **Document code:** A **CLC number:** TP311

INTRODUCTION

Various new database applications have been developed in many respects, such as data warehousing and content-based multimedia information retrieval. In these applications, one of the most frequently used but expensive operations is to find objects in databases that are similar to a given query object. Nearest neighbor search is a central requirement in such cases (Chavez *et al*., 2001). Since the databases are very large and consist of millions of data objects with several tens to a few hundreds of dimensions, it is essential to use appropriate multidimensional indexing techniques to achieve efficient search of the data.

There is extensive research on solving the nearest neighbor search problem, with a large number of multi-dimensional indexes being developed for this purpose. However, most of these structures are based on partitioning the data space (Fonseca and Jorge, 2003; Hjaltson and Samet, 2003). Because of the ‘curse of dimensionality’, these methods are out-performed on average by a simple sequential scan if the number of dimensions exceeds around 10 (Weber *et al*., 1998). To break this curse, many new proposals are presented, such as approximate vector presentation and 1D transformation. However, most of these proposals do not efficiently utilize the advantages of 1D transformation and data approximation and typically perform well for certain cases only (dimensionality, data distribution, query type, etc.).

In this paper, we propose a novel index structure for fast KNN search, called Bit-code and Distance based index (BD). The basic idea of BD is that a high-dimensional vector is first approximately represented and then transformed into a 1D value, with the key being managed by a B^+-tree. This is done using a two-step algorithm. In the first step, the high-dimensional data space is split into $2^{d'}$ ($d'<d$)

* Project (No. [2005]555) supported by the Hi-Tech Research and Development Program (863) of China

partitions, where d' is the number of dimensions on which the space is partitioned. Each partition can be represented by a unique bit string of length d', called bit code. A data point is then approximated by the bit code of the partition it falls into. In the second step, each data point is transformed into a 1D value, termed as key, by the distance with respect to the reference point. This allows the key to be indexed using a B^+-tree structure and KNN search to be performed using 1D range search. In the process of KNN search, with the bit code we can rapidly eliminate irrelevant partitions that are not intersected by the query without the expensive distance computation. Then, we can determine which key values inside an affected partition are affected by the query. Therefore, the BD can reduce both I/O and computation costs for KNN search.

We implemented the BD method together with a new KNN search algorithm, and conducted an extensive performance study to evaluate their effectiveness. Our results showed that the proposed schemas can provide fast response time without sacrificing the quality of the answers. Moreover, through appropriate choice of partition scheme, the BD method can compute the complete answer set much faster than the existing index structures on both real life and synthetic high-dimensional datasets.

The rest of this paper is organized as follows. In Section 2, we give an overview of the related work. In Section 3, we introduce and discuss our new technique, the BD. Then we present the experimental results and analyze the benefits of BD in Section 4. Section 5 concludes the paper.

RELATED WORK

Many index structures have been proposed for similarity search in high-dimensional spaces. Most of them are based on data space partitioning, which include the R^*-tree, the X-tree, the SR-tree, the TV-tree and many others. Although these methods generally perform well at low dimensionality, they suffer greatly from the curse of dimensionality (Beyer *et al*., 1999; Hinneburg *et al*., 2000). Recent proposals to tackle this problem can be categorized into three approaches: dimensionality reduction, data approximation, and 1D transformation.

Dimensionality reduction methods (Jin *et al*., 2003) map the high-dimensional space into a low-dimensional space which can be indexed efficiently using existing multi-dimensional indexing techniques. The main idea is to condense the original space into a few dimensions along which the information is maximized. Such methods yield approximate nearest neighbors, however, since dimensionality reduction incurs information loss.

Representations of the original data points using smaller and approximate representations have also been proposed as a means of aiding high-dimensional indexing and searching by applying the efficiency of sequential scan. There are two examples VA-file (Weber *et al*., 1998) and BID (Cui *et al*., 2005) that are related to our work. The VA-file represents the original data points by much smaller vectors, which are then sequentially scanned to obtain a very small set of candidates. However, the VA-file cannot adapt effectively to different data distributions, mainly due to its unified cell-partitioning scheme. Different from the VA-file, the BID employs one bit to represent each feature vector of a point and the number of bit-difference is used to prune away the further points, so that it can support approximate KNN search in the main memory environment. Although the BID speeds up the search with no distance computation, no guarantee can be given on the accuracy of the result.

One-dimensional transformations provide another direction for high-dimensional indexing (Berchtold *et al*., 1998; Yu *et al*., 2001; 2004; Zhang *et al*., 2004). PT (Berchtold *et al*., 1998) and iDistance (Yu *et al*., 2001) are such efficient methods. PT divides the d-dimensional data space into $2d$ pyramids and then cuts each pyramid into slices, each of which forms a data page. It allows mapping from d-dimensional space to single-dimensional space. However, PT is only highly adapted to hypercube-shaped window queries over uniform data, which limits its usefulness in real applications. The iDistance relies on clustering the data and indexing the distance of each data point to its corresponding reference point. The choice of partition and reference point provides the iDistance technique with degrees of freedom that most other techniques do not have. However, when dimensionality exceeds 30, the equal distance phenomenon occurs, so the effectiveness of pruning degenerates rapidly.

BIT-CODE AND DISTANCE BASED INDEX

In this section, we first present our encoding strategy and data compact representation for high-dimensional data points. We then present a KNN search algorithm that can be used to facilitate searching using the proposed technique. Finally, we discuss how the basic schema can be optimized.

Data representation

To support bit-code and distance based similarity search, we need a reference point, and partition the data space on some dimensions d' ($<d$). Based on the distribution of the data, we can determine a threshold value d'. We shall defer the discussion on d' to the subsection of "Tuning of d'". The choices of the reference point are as follows: the center of the space is selected if the data are uniformly distributed; for clustered data, we can group the data points into a set of clusters by existing techniques, such as K-means, BIRCH or CURE (Guha *et al.*, 1998), and hence the center of all the centroids is selected.

For the rest of this paper, we assume the data space to be the d-dimensional unit hyper-sphere, with the reference point $O(o_1, o_2, ..., o_d)$. Let $Dist(\cdot,\cdot)$ be a metric distance function for pairs of points. In our research, we use the Euclidean distance as the distance function, although other distance functions may be used for certain applications.

Definition 1 (Bit Code) The bit code $S(s_1, s_2, ..., s_{d'})$ of a random point $P(p_1, p_2, ..., p_d)$ is defined as

$$s_i = \begin{cases} 1, & p_i > o_i, \\ 0, & \text{otherwise,} \end{cases} \quad 1 \le i \le d'.$$

The intuitive meaning of Definition 1 is that the bit code S of a point P depends on its feature vectors on dimensions d', that is to say, the value of s_i is just the bit code of p_i. Therefore, a d-dimensional vector can be approximately represented by the bit code of the partition it falls into, with length d'. In the 2D example depicted in Fig.1, the space has been divided into 4 partitions, encoded by '00', '01', '10' and '11' respectively.

Definition 2 (1D Key) A data point $P(p_1, p_2, ..., p_d)$ with bit code $S(s_1, s_2, ..., s_{d'})$ has an index key as

$$Key_p = Value(S) \cdot C + Dist(P, O),$$

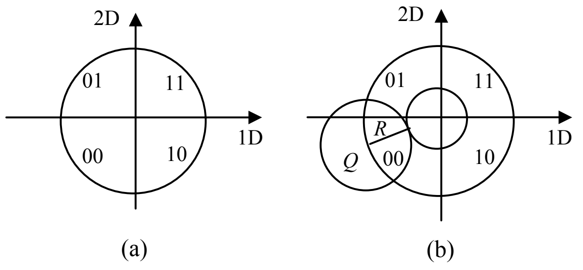

Fig.1 (a) Space partition in 2D space; (b) Search regions based bit code and distance

where $Value(S)$ is a function that returns an integer value of S, and $Dist(P,O)$ is the distance between P and O, and C is a constant to stretch the data range.

Note that, in Definition 2 the function $Value(S)$ can be defined as $Value(S)=2^0\cdot s_1+2^1\cdot s_2+\ldots+2^{d'-1}\cdot s_{d'}$. Potentially, any function can be used if they can keep all partitions apart. In the d-dimensional unit space, $Dist(P,O)$ has the maximum value $d^{1/2}$, such that $C=\lceil d^{1/2}\rceil$ can serve to partition the single dimension space into regions and all points in partition S will be mapped to the range $[Value(S)\cdot C, (Value(S)+1)\cdot C]$.

Given the transformation determining the key value of a point, it is a simple task to build an efficient index structure such as the B$^+$-tree. Note that although we index our data using a 1D key, we store d-dimensional points plus the corresponding 1D key in the leaf nodes of the B$^+$-tree.

KNN search

The algorithm for KNN search often begins by searching a small 'sphere', and incrementally enlarges the search space till all the K nearest neighbors are found. However, due to the variation of the position of the query point and the density of the data distribution, it is difficult to determine the radius of the beginning query sphere and its increment in the iteration process. For the BD, we present a new KNN search algorithm with the main idea as follows: We first obtain a KNN answer set as the initial query sphere within the partition the query point falls into; and then, we have to examine which partitions are affected by the query; and finally, we have to determine the ranges inside the partitions. Note that once the KNN result is modified, so is the query sphere. Moreover, those partitions close to the query are examined before the faraway partitions, which can be determined by their bit codes with respect to the partition the query falls into, so that the radius of the query sphere and the efficient range of the B$^+$-tree can be reduced as soon as possible. Fig.2 summarizes the algorithm for KNN search with the BD method.

```
Initialize S;   // answer set
Examine the partition that Q falls into;
Set the query sphere as center Q and radius the
   distance of the farthest object in S from Q;
L: determine the intersected partitions;
      //using Lemma 1
   determine the affected ranges; //using Lemma 2
Examine these ranges in a suitable order;
If S is modified then goto L;
```

Fig.2 The main search algorithm of BD KNN

Our KNN search algorithm has many advantages over the commonly used method mentioned above: (1) The repeated scanning of a branch of the B$^+$-tree is avoided. (2) According to the value of dimensions d' and the data distribution, it is most possible all the K nearest neighbors are in the same partition as the query point, so that the beginning radius of the query sphere cannot be too large. (3) The query sphere is tuned so appropriately that a query requires less searching to retrieve the exact KNN answers. (4) It can provide an approximate KNN result, which can be returned to the users immediately and refined as more accurate answers are obtained in the subsequent examination. In the 2D example depicted in Fig.1b, for the query range with center Q and radius R, it intersects several partitions with bit codes '10' and '11', with the shaded ranges having to be searched.

Lemma 1 (Intersection of a Partition and a Sphere) A partition $T(t_1, t_2, ..., t_{d'})$ is intersected by a hypersphere with center $Q(q_1, q_2, ..., q_d)$ and radius R if and only if

$$t_i = \begin{cases} s_i, & |q_i - o_i| \ge R, \\ 0 \text{ or } 1, & \text{otherwise,} \end{cases} \quad 1 \le i \le d',$$

where $S(s_1, s_2, ..., s_{d'})$ is the bit code of Q.

Proof The query sphere intersects partition $T(t_1, t_2, ..., t_{d'})$, iff there exists a point $P(p_1, p_2, ..., p_d)$ inside the sphere which falls into partition T. Thus the distance of all coordinates $|p_i-q_i|$ must be smaller than R if $i<d'$. More formally: $\forall i\in[1, d']$, $|p_i-q_i|\le R$.

For the value of q_i, there are two cases to consider:

Case 1 $|q_i-o_i|\ge R$

In this case if $q_i>o_i$, then $p_i>o_i$ always holds. It can be similarly derived that $p_i<o_i$ holds if $q_i<o_i$. Thus, P has the same bit code on the ith dimension as Q, formally $t_i=s_i$.

Case 2 $|q_i-o_i|<R$

In this case we have $p_i>o_i$ or $p_i<o_i$. It implies that the bit code of P on the ith dimension is not related to q_i.

Lemma 2 (Interval of Intersection of Query and Partition) For a search sphere with query point Q and search radius R, the intersection key interval within an affected partition $T(t_1, t_2, ..., t_{d'})$ is defined as

$$[Value(T)\cdot C+Dist(Q,O)-R, Value(T)\cdot C+Dist(Q,O)+R].$$

Tuning of d'

Next, we discuss how the threshold value d' can be determined. Clearly, different d' values should be used for different data distributions. The determination of d' affects the effectiveness of bit codes. The ideal scenario is to have a large d' and a large number of dimensions encoded in a bit code. A larger d' will lead to intolerable exponential number of partitions. However, a much smaller d' will incur much information loss. As a result, a trade-off has to be made.

For real world applications datasets are often skewed and some features are more important than the other features. We select d' by employing Principle Component Analysis (PCA) (Jolliffe, 1986). As we know, PCA is the best in the mean-square error sense, linear dimension reduction technique. There are as many as PCs as the number of the original dimensions. After PCA processing, most of the information in the original space is condensed into the first few dimensions along which the variances in the data distribution are the largest. As such, the first d' dimensions can be selected, on which the data space is split into $2^{d'}$ partitions. For this purpose, before constructing the BD structure, we have to transform the data into the principal component space.

For the number of dimensions d', we adopt a simple strategy: we estimate it based on the fan-out of a leaf node of the B$^+$-tree, e.g., given a set of N points, and a fan-out of f, the number of dimensions is $d'=\lfloor\log_2(N/f)\rfloor$. Intuitively, such selection indicates all the points in a partition forms a data page, which will reduce I/O cost for KNN search using our method. Moreover, it is effective in partitioning the space for the BD: (1) For a high-dimensional dataset, whether it has a large number of points or not, it can be partitioned effectively. (2) Dimensions used by partitioning are not related to the data distribution, which makes the BD adaptable to different data distributions.

EXPERIMENTAL EVALUATION

To demonstrate the practical impact of the BD and to verify our theoretical results, we performed an extensive experimental evaluation of the BD and compared it to the following competitive techniques: VA-file, iDistance, Sequential Scan. Our evaluation comprises both real and synthetic high-dimensional datasets. The synthetic dataset contains 300 000 uniformly distributed points in a 60-dimensional data space. The real dataset contains 32-dimensional color histograms extracted from 68 040 images. The performance is measured in terms of the CPU time and the average disk page access over 100 different queries. For each query, the number of nearest neighbours to search is 10 unless otherwise stated. All experiments are run on a computer with Pentium III 800 MHz CPU and 256 MB RAM. The page size is 4 kB.

Evaluation using synthetic data

In our first experiment, we measured the performance behavior with varying number of data points. We performed 10 NN queries over the 20-dimensional synthetic dataset and varied the data size from 50 000 to 300 000.

Fig.3 shows three index structures outperforming the Sequential Scan significantly. This is because sequential scan of a dataset entails examination of each data point and calculation of distance between each point and the query point, which will result in high CPU cost. Therefore, we focus on the comparison of the BD with the VA-file and iDistance in the following experiments.

It is evident that the BD outperforms the VA-file and iDistance in Fig.3: the BD achieves a speedup factor of 18 over the iDistance, and 22.6 over the VA-file. It indicates that the BD has the properties of

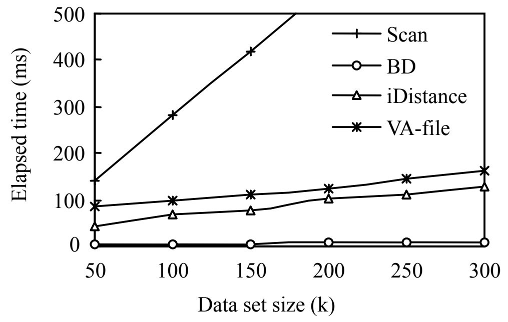

Fig.3 Performance of the four index structures over data size when *dim*=20, *K*=10

both the VA-file that accelerates sequential scan by the use of approximate vector, and the iDistance that reduces the range to search by the use of 1D vector. Additionally, for the BD, a fast KNN search algorithm is employed to reduce the search cost of the B^+-tree, so that the CPU cost can be improved.

In the second experiment, we determine the influence of the data space dimension on the performance of KNN queries. For this purpose, we created six 100k synthetic datasets with dimensionality 10, 20, 30, 40, 50 and 60 to run our experiments. Fig.4 shows that the CPU time using any index structure grows slowly with increasing dimension. However, a comparable deterioration of the performance with increasing dimension is not observable when using the BD. The experiment yields a speedup factor of up to 122.9 over the iDistance, and 182.4 over the VA-file. The increasing dimensionality theoretically brings more benefits to the BD because the data distribute sparsely and the distance to the nearest neighbor approaches the distance to the farthest neighbor. The BD split the high-dimensional space only on some dimensions determined by PCA such that it has an

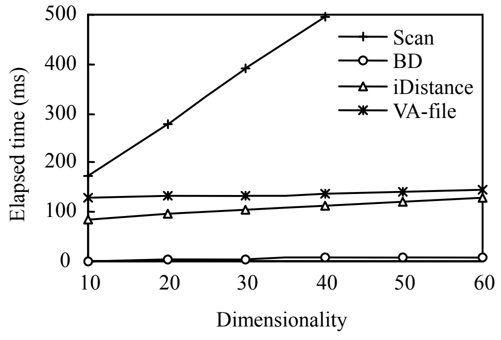

Fig.4 Performance of the four index structures over dimension when *N*=100 000, *K*=10

efficient pruning. On the other hand, the VA-file cannot make full use of the skewing characteristics, which makes a large amount of cells to search. By indexing the distance of each point to the nearest reference point, the iDistance may introduce many false hits during KNN search with the effectiveness of pruning degenerating rapidly as dimension increases. Therefore, according to the characteristics of the data distribution in high-dimensional spaces, the BD has a smarter partition schema.

Evaluation using real dataset

In this series of experiments, we used the real dataset extracted from 68040 pixel images. The effects of increasing values of K in KNN search are tested. Fig.5 shows the experimental results when K ranges from 10 to 40. The performance of the VA-file and iDistance is pretty close to each other. The BD still remains a good performance. The smarter partitioning schema and the pruning effectiveness of the BD help it to benefit more from the skewness of the color histograms.

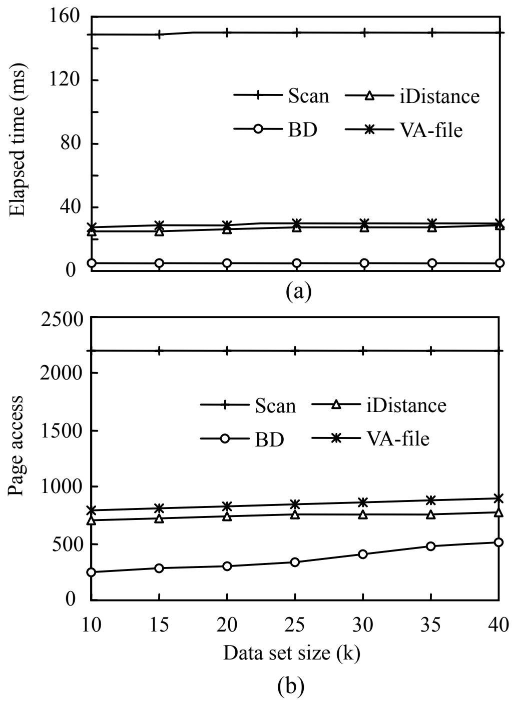

Fig.5 Performance of the four index structures over K when N=68040, dim=32. (a) Elapsed time; (b) Page access

CONCLUSION

In this paper, based on approximate vector presentation and 1D transformation, we proposed a new index structure, the BD. It is based on a special partitioning schema which is optimized for high-dimensional KNN queries, so that the BD can be adaptable to different data distributions. By indexing the bit-code and distance, most irrelevant data points can be eliminated rapidly in the process of KNN search. Furthermore, according to its characteristics, a fast KNN algorithm is presented with many benefits. Extensive experiments showed that the BD can achieve better performance than many other indexing techniques.

References

Berchtold, S., Bohm, C., Kriegel, H.P., 1998. The Pyramid-Technique: Towards Breaking the Curse of Dimensionality. Proc. ACM SIGMOD Int. Conf. on Management of Data, p.142-153.

Beyer, K., Goldstein, J., Ramakrishnam, R., 1999. When is "Nearest Neighbor" Meaningful? Proc. 7th Int'l Conf. Database Theory, p.1-11.

Chavez, E., Navarro, G., Baeza-Yates, R., Marroquin, J., 2001. Searching in metric spaces. *ACM Computing Surveys*, **33**(3):273-321. [doi:10.1145/502807.502808]

Cui, B., Shen, H.T., Shen, J., Tan, K.L., 2005. Exploring Bit-Difference for Approximate KNN Search in High-dimensional Databases. Proc. 16th Australian Database Conference, p.165-174.

Fonseca, M.J., Jorge, J.A., 2003. Indexing High-dimensional Data for Content-based Retrieval in Large Databases. Proc. 8th International Conference on Database Systems for Advanced Applications, p.267-274.

Guha, S., Rastogi, R., Shim, K., 1998. Cure: An Efficient Clustering Algorithm for Large Databases. Proc. ACM SIGMOD Int. Conf. on Management of Data, p.73-84.

Hinneburg, A., Aggarwal, C.C., Keim, D.A., 2000. What is the Nearest Neighbor in High-dimensional Spaces. Proc. 26th Int. Conf. on Very Large Data Bases, p.506-515.

Hjaltson, G.R., Samet, H., 2003. Index-driven similarity search in metric spaces. *ACM Trans. on Database Syst.*, **28**(4):517-580. [doi:10.1145/958942.958948]

Jin, H., Ooi, B.C., Shen, H.T., Yu, C., Zhou, A.Y., 2003. An Adaptive and Efficient Dimensionality Reduction Algorithm for High-dimensional Indexing. Proc. Int'l Conf. Data Eng., p.87-98.

Jolliffe, I.T., 1986. Principal Component Analysis. Springer-Verlag, New York.

Weber, R., Schek, H.J., Blott, S., 1998. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-dimensional Spaces. Proc. 24th Int. Conf. on Very Large Data Bases, p.194-205.

Yu, C., Ooi, B.C., Tan, K.L., Jagadish, H.V., 2001. Indexing the Distance: An Efficient Method to KNN Processing. Proc. 27th Int. Conf. on Very Large Data Bases, p.421-430.

Yu, C., Bressan, S., Ooi, B.C., Tan, K.L., 2004. Querying high-dimensional data in single dimensional space. *Int. J. Very Large Data Bases*, **13**(2):105-119.

Zhang, R., Ooi, B.C., Tan, K.L., 2004. Making the Pyramid Technique Robust to Query Types and Workloads. Proc. Int. Conf. on Data Eng., p.313-324. [doi:10.1109/ICDE.2004.1320007]