Maximal Metric Margin Partitioning for Similarity Search Indexes

Hisashi Kurasawa
The University of Tokyo
2-1-2 Hitotsubashi,
Chiyoda-ku, Tokyo, JAPAN
kurasawa@nii.ac.jp

Daiji Fukagawa, Atsuhiro Takasu,
Jun Adachi
National Institute of Informatics
2-1-2 Hitotsubashi,
Chiyoda-ku, Tokyo, JAPAN
{daiji,takasu,adachi}@nii.ac.jp

ABSTRACT

We propose a partitioning scheme for similarity search indexes that is called Maximal Metric Margin Partitioning (MMMP). MMMP divides the data on the basis of its distribution pattern, especially for the boundaries of clusters. A partitioning surface created by MMMP is likely to be at maximum distances from the two cluster boundaries. MMMP is the first similarity search index approach to focus on partitioning surfaces and data distribution patterns. We also present an indexing scheme, named the MMMP-Index, which uses MMMP and small ball partitioning. The MMMP-Index prunes many objects that are not relevant to a query, and it reduces the query execution cost. Our experimental results show that MMMP effectively indexes clustered data and reduces the search cost. For clustered vector data, the MMMP-Index reduces the computational cost to less than two thirds that of comparable schemes.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—*Multimedia databases*; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Indexing methods*

General Terms

Algorithms, Performance

Keywords

Similarity search, Indexing, Metric space

1. INTRODUCTION

A similarity search efficiently finds objects that are similar to a query from a large dataset [10]. Similarity searches based on a metric space can be applied to all types of data whose distances obey metric space postulates such as the triangle inequality. Therefore, metric space indexes are very useful for applications that deal with huge amounts of vectors, strings, graphs, tags, and so on.

Many similarity search indexes use *Pivot Partitioning*. A *Pivot* is a reference object in an index. Pivot partitioning divides the space into regions based on the distance from the pivot. When searching for objects, the regions far from the query are pruned. Its performance depends on the pivot selection. A better division prunes more objects during the search. Early pivot selections were based on heuristics [9]. They use simple statistical features such as the mean and the variance for selecting pivots. However, their choices of pivots are only a little better than random selection. Recently, other selection mechanisms have been proposed on the basis of data distribution [6]. These methods set cluster centers as the pivots and divide the data on the basis of the distances from the pivots. Although they can effectively classify dense regions, they only work well on particular distribution patterns. A cluster may be separated by pivots, because their partitioning surfaces are based on cluster centers rather than cluster shapes.

We propose a novel method for pivot partitioning called Maximal Metric Margin Partitioning (MMMP), which can be adapted to the shapes of data clusters. First, MMMP constructs hierarchical clusters with arbitrary shapes by using OPTICS [2]. After that, it divides the data on the basis of the boundaries of same-degree clusters from the root of the cluster branches. During this division, MMMP looks for the pivot object that maximizes the distances between the clusters. The main contribution of this paper is that we have developed a pivot selection for metric spaces that is based on maximal margins, that is effective for clustered data.

In this paper, we explain how MMMP divides data for a similarity search index. We also present an indexing scheme called MMMP-Index. We conducted two experiments. The first experiment evaluated the partitioning performance of MMMP. The other evaluated the indexing performance in comparison with existing schemes.

2. RELATED WORK

Many metric space indexing schemes use pivot partitioning. The early studies selected pivots that were far away from the other objects and other pivots [9]. Their aim was to get various distances from the pivot to each object and reduce useless pivots. Some also tried to exclude dense regions [5, 8]. These methods used simple statistical features such as the mean [3, 5], the variance [9, 8], and the ratio [7].

Recently, more elaborate features that are based on data distribution have been used [6]. iDistance clusters the search space and identifies the cluster centers. It uses a Voronoi partition, and it sets the cluster centers as pivots. This scheme divides the search space into regions by using the pivots representing the data clusters. This helps reduce the number of regions accessed during searching. A more recent method improves the indexing performance [11].

The above schemes have difficulty in handling objects in a sparse space because the clustering result would be meaningless in such a case. List of Clusters (LC) [4] is another way of exploiting the knowledge of the data distribution. It divides the search space by using small ball partitioning, and it recursively excludes the indexed objects from the space. The radius of the partitioning is determined from the number of objects inside the partition. However, it needs more pivots because of its smaller radius than other schemes.

We think pivot partitioning can further reduce the search cost by utilizing the data distribution of the objective dataset. In the existing approaches, skewed data clusters may be separated into multiple regions by pivots, because their partitioning surfaces are based on cluster centers rather than cluster shapes. This causes the an increase in query execution cost. Therefore, we developed a new partitioning scheme that is based on the shapes of the data clusters.

3. MMMP

This section proposes our pivot selection method, which we call *Maximal Metric Margin Partitioning (MMMP)*, that selects pivots based on the shapes of the data clusters.

For a metric space $M = (D, d)$, suppose a pivot p divides a set S of objects in D into two regions:

$$S_1 = \{o \in S \mid d(o,p) \le r_p\}\ , \qquad S_2 = \{o \in S \mid d(o,p) > r_p\}\ ,$$

where r_p is the partitioning distance for pivot p. When searching for objects within r_q from a query object q in terms of the metric space M, if the inequality

$$d(q,p) + r_q \le r_p\ , \tag{1}$$

holds, it is sufficient to check for objects in S_1. Similarly, if the inequality

$$d(q,p) - r_q > r_p\ , \tag{2}$$

holds true, it is sufficient to check objects within S_2. If the margin between regions S_1 and S_2 is large, either the inequality of (1) or (2) probably holds, i.e., we can prune region S_2 or S_1. On the other hand, if the boundaries between S_1 and S_2 are close to each other, we need to check the objects within both S_1 and S_2 for a query around the boundaries, even when the centers of S_1 and S_2 are far from each other. This leads us to conduct research to come up with a pivot selection method that is based on a large margin criterion.

To find pivots that divide the space into regions with a large margin, we first make hierarchical clusters that recursively divide the space into two regions. Currently, we use OPTICS [2], which is a density-based clustering method, in this step. To handle a large dataset, we use OPTICS on a randomly sampled dataset. Then, for each division, we find a pivot that separates the clusters with a large margin.

3.1 OPTICS

OPTICS is a kind of density-based Clustering [2]. OPTICS constructs hierarchical clusters with arbitrary shapes.

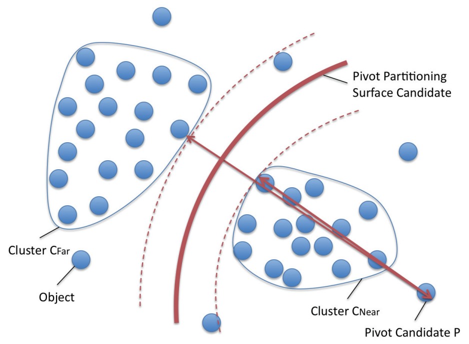


Figure 1: Pivot Selection in the MMMP

It automatically determines the number of clusters. The most important reason that we use OPTICS is to extract arbitrarily-shaped clusters while using only the distances between objects. Thus, other schemes such as k-means and BIRCH are not appropriate for MMMP. Moreover, the reason we require a hierarchical scheme is to order the cluster branches from a large margin one to small ones. That is, it executes a rough indexing.

3.2 Pivot Selection and Partitioning

MMMP aims to achieve effective search pruning. We focus on the partitioning surfaces and the data distribution patterns to achieve this requirement. We maximize the distances between the partitioning surface and its closest objects, because the objects can be clearly classified. We divide the space on the basis of the OPTICS's clustering results because it effectively separates small dense regions. Therefore, we try to find partitioning surfaces that are between pairs of cluster boundaries that are at maximum distances from the cluster boundaries. However, we cannot directly detect the partitioning surfaces in a metric space, because the partitioning surfaces are created by pivots. Thus, MMMP basically searches for a pivot object that divides the clusters obtained by OPTICS with the largest margin.

From the clustering results determined by using OPTICS, we construct hierarchical clusters represented by a binary tree. Each node in the tree corresponds to a subspace and it is further divided into two subspaces represented by its child nodes. For each node v in the tree, we extract a pivot in the following way. For an object p, let C_{Near} (resp. C_{Far}) denotes the v's child cluster that is closest to (resp. far from) p. Then, MMMP chooses the following object as a pivot

$$\mathrm{pivot} = \operatorname*{argmax}_{p \in S} \left(\min_{o_f \in C_{\mathrm{Far}}} d(p, o_f) - \max_{o_n \in C_{\mathrm{Near}}} d(p, o_n) \right)\ . \tag{3}$$

The first term in the right hand side of this formula represents the distance to the nearest object in C_{Far}, whereas the second term is the distance to the farthest point in C_{Near}. Therefore, the right hand side of Eq. (3) represents a kind of margin. Even if a cluster is partitioned into more than two regions in a cluster branch, MMMP merges them and then divides it into two regions. In that case, all the region combination cases are considered, and the highest scoring candidate is selected.

The reason why only one object in each cluster is used for distinguishing between C_{Near} and C_{Far} is as follows. One reason is to reduce the computational cost for this operation. The other reason is that the pivot candidates evaluated with the incorrect label clusters have no influence on the pivot selection. These pivot candidates are not relevant to the best pivot in most cases, and would finally be removed at Eq (3). Of course, when there is no relevant pivot candidate in the data, this distinction does not work well and the evaluation of a pivot may be a minus value. The partitioning distance Distance(p) of the pivot is defined as

$$\text{Distance}(p) = \frac{1}{2}\left(\min_{o_f \in C_{\text{Far}}} d(p, o_f) + \max_{o_n \in C_{\text{Near}}} d(p, o_n)\right) . \tag{4}$$

Fig. 1 shows the image of the pivot selection in the MMMP.

3.3 MMMP-Index

Our indexing scheme, named MMMP-Index, uses MMMP and small ball partitioning. As we described in the preceding section, MMMP is designed to prune regions irrelevant to the query in a clustered data space. However, MMMP does not work well when the number of clusters in the data space is too small or the size of any one cluster is too large. The number of objects managed by a pivot is related to the performance of the index. Therefore, we make the MMMP-Index divide a large cluster into small enough regions to be controlled by pivots like LC [4].

The index is implemented by two B+-trees. One B+-tree manages two kinds of pivots; the pivots selected by MMMP and those chosen by small ball partitioning. The pivots form a tree structure. Each pivot has its own ID, and hold the IDs of its leaf pivots. The smaller ID is assigned to the earlier selected pivot. Their keys are also based on their IDs. The other B+-tree stores objects. The object key is measured by $d(obj, p) + ID_p \times c$, where p is the pivot that manages the object and c is a parameter that is sufficiently larger than the distance between objects. With these key definitions, when we search for objects whose distances from the pivot are within a certain range, we can sequentially access the disk and can reduce the page access cost.

3.4 Index Construction Cost

The MMMP-Index is constructed by (a) the hierarchical clustering, (b) the pivot selection for each partitioning of a cluster in the hierarchy, and (c) the ball partitioning of the leaf clusters. Step (a) requires a $O(n \log n)$ to $O(n^2)$ calculation by OPTICS [2]. Step (b) requires $O(n^2)$, and step (c) does from $O(m \log m)$ to $O(m^2)$, where m is the size of the maximum leaf cluster [4]. Generally m is much less than n, so steps (a) and (b) are the dominant parts of the indexing construction. On the other hand, LC requires from $O(n \log n)$ to $O(n^2)$ [4]. So, the computational complexity for constructing the MMMP-Index is the same as the worst case of LC. Actually the indexing time for MMMP-Index was almost the same as that for LC in our experiments.

4. PERFORMANCE EVALUATION

We conducted two experiments evaluated the partitioning performance of MMMP and the indexing performance.

4.1 Data Set

The datasets consisted of synthetic vectors generated by us, and real vector datasets named Corel Image Features downloaded from the UCI KDD Archive [1]. We generated 2, 8, and 16-dimensional clustered vectors for the evaluation. The cluster centers of the clustered vectors were randomly selected. The numbers of the centers were 10, 20, and 30. The number of objects was randomly chosen for each cluster. The objects in the cluster were based on the normal distribution. Its standard deviations were randomly set from 0 to 0.10 and from 0 to 0.20. According to the previous studies [4], the size of the data were 100,000. Queries were randomly chosen from the same distribution as the data set. When the chosen query happened to be the same point in the data set, we discarded it. The ranges of a query were the distances to the k nearest neighbor objects. k ranged from 5 to 100.

4.2 Partitioning Performance

MMMP selects a pivot on the basis of the cluster shapes because the partitioning surface of the pivot was in a sparse space. We compared MMMP with D-Index [5] and iDistance [6]. D-Index selects a pivot on the basis of the mean distances between the pivot and objects, and recursively divides the space using Excluded Middle Partitioning. iDistance divides the space with a Voronoi partition based on k-means clustering, and sets the cluster centers as the pivots. To improve the search pruning and reduce the search cost, a query should refer to a smaller number of regions partitioned by the pivots. Thus, we evaluated these costs at the points of the accessed regions for the query. The clustering in MMMP was computed using 20,000 random samples from the dataset. The number of clusters in iDistance was set from 4 to 64. The reason we present so many instances of iDistance is that it cannot define an appropriate number of clusters from the data, and the cluster number was a very important factor in this experiment. The partitioning distance parameter in D-Index was set according to the shortest search response time. Each result was the average over 1,000 queries of its dataset.

Fig. 2 shows the results. The vertical axes represent the number of accessed regions. The horizontal axis is the query ranges. The number for each line in the legend represents the parameter for indexing. The numbers for MMMP and iDistance are the numbers of clusters, and the numbers for the D-Index are the partitioning distances of the exclusion sets. The results show that the MMMP minimizes the number of regions over which a query is in every data. The number of accessed regions was approximately one from the MMMP results.

Although the results seem to be sufficient enough for the partitioning evaluation, it is unclear whether MMMP effectively divides the space for search pruning. We need to find out whether the sizes of the regions are appropriate. To give an example of a bad situation, if one region is very large and the others are very small, a query may be over only the large region. However, it is difficult to see if this is the case, because the standard deviations and the number of objects in each cluster of the datasets are randomly selected. Therefore, we conducted another experiment to evaluate the indexing performance. We show in Sec. 4.3 how the MMMP reduces the page access cost and the computational cost by pruning the search.

4.3 Index Performance

We conducted experiments to evaluate the MMMP-Index together with MMMP. We compared the MMMP-Index with iDistance [6], D-Index [5], and LC [4]. As in the related work

[10], the indexes' performances were evaluated in terms of the page access and computational costs. The parameters of each scheme were set on the basis of the shortest search response time. According to the previous studies [6], the page size needed for estimating the page access cost was 4096 bytes. Each result was the average over 1,000 queries of its dataset.

In each figure, the red line is the MMMP-Index, the green line is LC, the blue line is D-Index, and the pink line is iDistance. It is clear that the MMMP-Index is superior to the other schemes for indexing clustered data. To take the results for 8-dimension clustered vectors as an example, the computational cost of the MMMP-Index is less than two thirds that of the other schemes. Comparing Fig. 3 with Fig. 4, the computational cost of the MMMP-Index for the clustered data with 20 clusters is almost half that of the data with 10 clusters. The page access cost of the MMMP-Index is also less than the other schemes. The results in Sec. 4.2 and in this section prove that MMMP is useful for clustered vectors. MMMP achieves effective partitioning by exploiting knowledge about the data distribution.

5. CONCLUSION

We developed MMMP as a means of pivot partitioning for the purpose of pruning in a similarity search index. MMMP divides data based on the cluster shapes extracted by using OPTICS. During partitioning, MMMP looks for an object to be the best pivot whose partitioning surface maximizes the distances from the cluster boundaries. We also created an index, named the MMMP-Index. MMMP is most effective when the indexed data is clustered. We are currently working on reducing the pivot selection cost.

6. REFERENCES

[1] Uci kdd archive, http://kdd.ics.uci.edu/.

[2] M. Ankerst, et al. Optics: ordering points to identify the clustering structure. In *SIGMOD*, 1999.

[3] B. Bustos, et al. Pivot selection techniques for proximity searching in metric spaces. *Pattern Recognition Letters*, 24(14):2357–2366, 2003.

[4] E. Chevez et al. A compact space decomposition for effective metric indexing. *Pattern Recognition Letters*, 24(9):1363–1376, 2005.

[5] V. Dohnal, et al. D-index: Distance searching index for metric data sets. *Multimedia Tools and Applications*, 21(1):9–33, 2003.

[6] H. V. Jagadish, et al. idistance: An adaptive b+-tree based indexing method for nearest neighbor earch. *ACM Trans. on Database Systems*, 30(2):364–397, 2003.

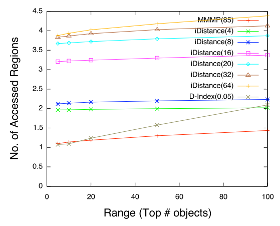
[7] O. Pedreira et al. Spatial selection of sparse pivots for similarity search in metric spaces. In *SOFSEM*, 2007.

[8] J. Venkateswaran, et al. Reference-based indexing of sequence databases. In *VLDB*, 2006.

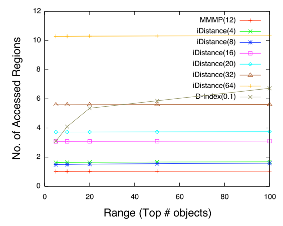
[9] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *SODA*, 1993.

[10] P. Zezula, et al. *Similarity Search: The Metric Space Approach*. Springer-Verlag New York, Inc., 2005.

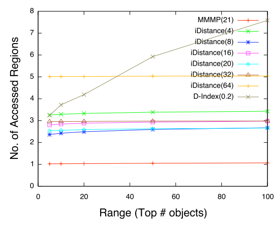
[11] Y. Zhuang, et al. Indexing high-dimensional data in dual distance spaces: a symmetrical encoding approach. In *EDBT*, 2008.



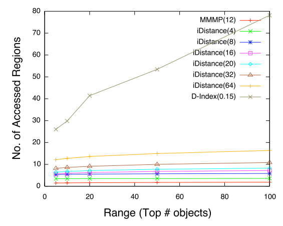
(a) 2D, 20 clusters,σ:(0,0.10)



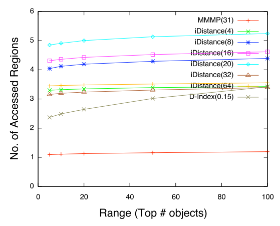
(b) 8D, 10 clusters,σ:(0,0.10)



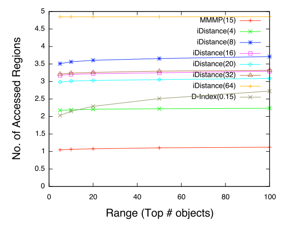
(c) 8D, 20 clusters,σ:(0,0.10)



(d) 8D, 20 clusters,σ:(0,0.20)

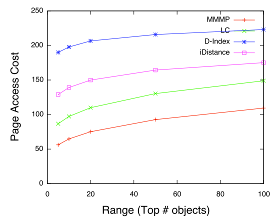


(e) 8D, 30 clusters,σ:(0,0.10)

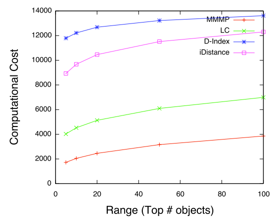


(f) 16D,20 clusters,σ:(0,0.10)

Figure 2: Partitioning Performance

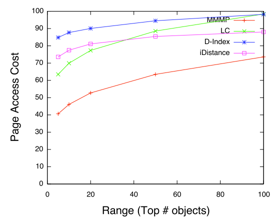


(a) Page Access Cost

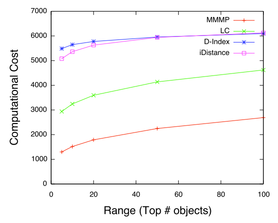


(b) Computational Cost

Figure 3: 8D, 10 clusters, σ:(0,0.10)

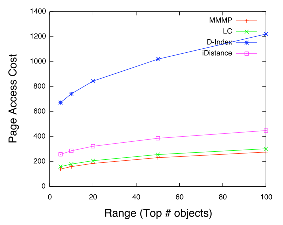


(a) Page Access Cost

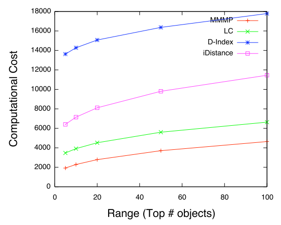


(b) Computational Cost

Figure 4: 8D, 20 clusters, σ:(0,0.10)



(a) Page Access Cost



(b) Computational Cost

Figure 5: Corel Image Features