Chapter 2

Approximate Retrieval of High-dimensional data with L_1 Metric by Spatial Indexing

This chapter describes a method of approximate retrieval in L_1 metric space.

2.1 Introduction

Currently, multimedia data retrieval systems are mostly similarity-based. The common method is to extract the features (usually in the form of a vector) from each data in the database and then to map features into points in a multidimensional feature space. The distance between two feature points is frequently used as a measure of similarity between the two corresponding multimedia data. Once the distance or similarity function is defined

for the multidimensional feature space, an approximate retrieval can be used to retrieve the data that satisfy the criteria specified in a given query.

Many approaches of the similarity search assume that A suitable distance function is known a priori. Euclidean distance is usually used for measurement[17]. However, it is not trivial to define a distance function that best mimics human visual perception regarding multimedia object similarity measurements. We propose a technique of approximation retrieval in L_1 metric space.

We denote the sets of natural numbers and real numbers by $\mathcal{N}$ and $\mathcal{R}$, respectively. Let S be a finite set of objects $\{O_1, O_2, \ldots, O_m\}$ and $D : S \times S \rightarrow \mathcal{N}$ be a function which gives the distance between objects. We call (S, D) an *object space.* A *query* is given as a pair (Q, h) of an object $Q \in S$ and a natural number h. The *answer* $Ans(Q, h)$ to a query (Q, h) is the set of objects within distance h from Q, that is,

$$Ans(Q, h) = \{O_i \in S \mid D(Q, O_i) \leq h\}.$$

The above setting of approximate retrieval as $Ans(Q, h)$ is very natural and general. When (S, D) is a Euclidean space, most spatial indexing structures are almost directly used to realize approximate retrieval. In many cases, however, unless objects are inherently geometrical like map information, object space is not Euclidean.

Here, we assume (S, D) can be considered as a metric space based on discrete L_1 (or,

Manhattan) distance, that is,

$$S \subseteq \mathcal{N}^n \text{ and } D(O_i, O_j) = \sum_{k=1}^{n} |O_i^{(k)} - O_j^{(k)}|,$$

where $O_i^{(k)}$ and $O_j^{(k)}$ are the k-th coordinates of objects O_i and O_j, respectively. Most of the difference measures might be captured as a discrete L_1 distance. For example, a natural definition of distance between objects consisting of several attribute values may be the sum of the symmetric differences between each attribute values. This definition can be applied to many sort of objects, such as, documents, digital images, and game boards.

We adopt R-tree [7, 23] as a spatial indexing/access method. As Chakrabarti pointed out [13], R-tree can efficiently be used only for relatively low-dimensional objects. Therefore, we have to map high-dimensional objects into a lower dimensionality. We can use the FastMap method [18] by Faloutsos and Lin to project objects in Euclidean space into a lower dimensional space. Since FastMap is based on orthogonal projection in Euclidean space, we have to embed objects into a Euclidean space. However, L_1 distance cannot be embedded into any Euclidean space, in general. As we will see in Section 2.2, if we take the square root of L_1 distance as the distance, the objects can be embedded into a Euclidean space. In other words, if we define

$$D^{\frac{1}{2}}(X, Y) = \sqrt{D(X, Y)},$$

$(S, D^{\frac{1}{2}})$ can be embedded into a Euclidean space. If we appropriately map objects to

vertices of unit n_0-cube, then the Euclidean distance between vertices coincides with the square root of the L_1 distance between objects.

Here, we briefly explain the FastMap method. Consider a set of objects $\{O_1, O_2, \ldots, O_m\}$ in a Euclidean space, where $d(O_i, O_j)$ gives the Euclidean distance between objects O_i and O_j. Let take arbitrarily a pair (O_a, O_b) of objects, which is called a *pivot.* The first coordinate X_i of an object O_i is given by

$$X_i = \overline{O_a E} = \frac{(d(O_a, O_i))^2 + (d(O_a, O_b))^2 - (d(O_b, O_i))^2}{2d(O_a, O_b)}$$

where E is the image of O_i by the orthogonal projection to the straight line O_aO_b (Fig. 2.1). Here, we should note that distances between objects are enough to calculate the coordinate X_i and any coordinates of objects are not necessary. Let O_i' be the image of O_i by the orthogonal projection to the hyper-plane that is orthogonal to the straight line O_aO_b. The distance between O_i' and O_j' is given by

$$(d(O_i', O_j'))^2 = (d(O_i, O_j))^2 - (X_i - X_j)^2.$$

Thus, we can repeatedly apply the above projection to get the second and other coordinates of objects. One of the most important issues in applying FastMap may be how to select pivots. Intuitively, the better pivot should provide the more selectivity in retrieval. Details are discussed by Faloutsos and Lin [18].

Let p be an orthogonal projection to $\mathcal{R}^{k_0}$ which is obtained by FastMap, where (S, D) is

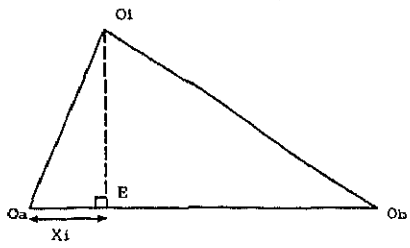

Figure 2.1: Orthogonal projection to pivot line

the original object space and $D^{\frac{1}{2}}$ is used as the distance function in applying FastMap. We call $\mathcal{R}^{k_0}$ the *index space*. Since p is an orthogonal projection, the distance between images of objects in the index space is not larger than the square root of the distance between objects, that is, $d(p(O_i), p(O_j)) \leq D^{\frac{1}{2}}(O_i, O_j)$. For a query (Q, h), we have

$$\{O_i \in S \mid d(p(Q), p(O_i)) \leq \sqrt{h}\} \supseteq Ans(Q, h).$$

Therefore, we can retrieve all the necessary objects even after reducing dimension by FastMap. Such a retrieval is easily realized by using spatial access method like R-tree. The result from the method may include irrelevant objects to the query, which is caused by FastMap projection. To get exact answer, screening might be needed.

From the experiments of our method, we observed that the image of the query range in the index space $\mathcal{R}^{k_0}$, which is naturally considered as a $k0$-sphere with radius $h^{\frac{1}{2}}$, is too large to get all the necessary objects. Precisely, we can prove

$$\{O_i \in S \mid |p^{(k)}(Q) - p^{(k)}(O_i)| \leq \lambda_k h \text{ for all } k = 1, \ldots, k_0\} \supseteq Ans(Q, h),$$

where $p^{(k)}(O)$ is the k-th coordinate of the image of O in the index space and λk is a constant which is usually much smaller than 1. Thus, the query range of k_0-box, which

is smaller than the k_0-sphere, is enough to retrieve the correct answer. This phenomenon, which is derived from the combination of our object embedding into unit n_0-cube and FastMap, will be theoretically explained as the contraction of query range by FastMap in Section 2.3.

2.2 Embedding L_1 distance into Euclidean space

Theorem 1 *For any object space* (S, D), $(S, D^{\frac{1}{2}})$ *can be embedded into Euclidean space.*

Proof Without loss of generality, we assume that $S = \{O_1, \ldots, O_m\} \subseteq \mathcal{N}^n$. For each $k = 1, \ldots, n$, we use a bit vector of length b_k where $b_k = \max\{O_i^{(k)} \mid i = 1, \ldots, m\}$ and map the k-th coordinate value v to $u_{b_k}(v) = 1^v 0^{b_k - v}$, which is a bit vector such that the first v bits are 1 and other bits are 0. Here we identify bit vectors and bit strings. For each object O_i, we map O_i to a bit vector $u(O_i) = u_{b_1}(O_i^{(1)}) u_{b_2}(O_i^{(2)}) \cdots u_{b_n}(O_i^{(n)})$. Clearly, for any $k \in \{0, \ldots, n\}$ and any $v, v' \in \{0, \ldots, b_k\}$, $(d(u_{b_k}(v), u_{b_k}(v')))^2 = |v - v'|$. Therefore, $d(u(O_i), u(O_j)) = D^{\frac{1}{2}}(O_i, O_j)$. Thus, we can embed (S, D) into unit n_0-cube, where $n_0 = b_1 + \cdots + b_n$. ∎

For example, consider points $O(0,0)$, $A(0,1)$, $B(0,2)$, and $C(1,1)$ in x-y plane as in Fig. 2.2. Distances between these points based on L_1 metric are $D(O,A) = D(A,B) = D(A,C) = 1$ and $D(O,B) = D(O,C) = D(B,C) = 2$. As long as using D as the metric, A should be on the straight line OB and O, B, and C should make a regular triangle no matter what embedding is used. The height of regular triangle OBC is the square root of 3, which contradicts to $D(A,C) = 1$. Thus, there is no Euclidean space where these

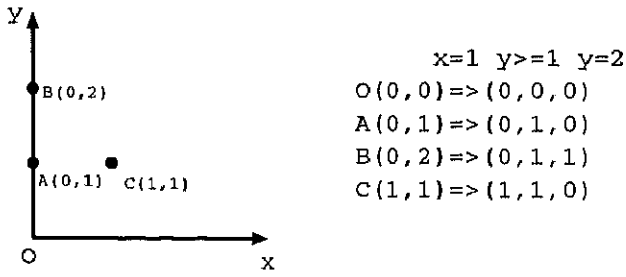

Figure 2.2: Embedding L_1 distance into unit n_0-cube

four points are embedded keeping the metric as it is. The maximum values of x and y coordinates are 1 and 2, respectively. Therefore, we map each point to a bit vector of length $n_0 = 1 + 2 = 3$. We can regard the first bit as representing if the x coordinate is equal to 1, the second as if the y coordinate is greater than or equal to 1, and the third as if the y coordinate is equal to 2. Clearly Euclidean distances between bit vectors are equal to respective L_1 distances between points in x-y plane.

From Theorem 1, we can apply FastMap to $(S, D^{\frac{1}{2}})$, which is embedded in Euclidean space of n_0-dimension. Here we should note that only distances between objects are sufficient for applying FastMap and actual values of coordinate in the Euclidean space are not necessary. Thus, the dimension n_0 of the Euclidean space, which may be quite larger than that of the original object space, does not the matter when we use FastMap.

2.3 Contraction of Query Range by FastMap

From Theorem 1, we can assume that every object is a vertex of a unit n_0-cube and the value of each coordinate is 0 or 1. As shown in Figure 2.3(A), let $\vec{P}_0$ be the vector between two objects used as the first pivot for FastMap. Let $\vec{e}$ be a unit vector of any coordinate in

$\mathcal{R}^{n_0}$. Then the length of the image of $\vec{e}$ by orthogonal projection to the first pivot is given by

$$\frac{|\vec{e} \cdot \vec{P}_0|}{|\vec{P}_0|}$$

Since every component of $\vec{P}_0$ is either -1, 0, or 1, the inner product $\vec{e} \cdot \vec{P}_0$ is also -1, 0, or 1. Therefore,

$$\frac{|\vec{e} \cdot \vec{P}_0|}{|\vec{P}_0|} \leq \frac{1}{|\vec{P}_0|}$$

Let define λ_1 as the right side of the above inequation. Consider two objects O_1 and O_2 such that $D(O_1, O_2) = h$ and a vector $\vec{v}$ between O_1 and O_2. Clearly, exactly h components of $\vec{v}$ are -1 or 1 and all the other components are 0. Therefore, the length of the image of $\vec{v}$ is less than or equal to $h\lambda_1$. Since $|\vec{P}_0|$ is usually larger than 1, λ_1 is relatively small. For the second and other projections by FastMap, similar phenomena can be derived.

Because the pivots($\vec{P}_0$, $\vec{P}_1$, ..., $\vec{P}_k$) are not perpendicular to each other, from the second pivot $\vec{P}_1$, the pivots should be projected on the hyperplane consisting of previous pivots. Assume H_0 is the hyperplane perpendicular to $\vec{P}_0$, $\vec{P}_1'$ is projection of $\vec{P}_1$ on the hyperplane H_0. Analogically, H_1 is the hyperplane perpendicular to $\vec{P}_1'$(Note; not $\vec{P}_1$) in hyperplane H_0, and $\vec{P}_2'$ is projection of $\vec{P}_2$ on the hyperplane H_1. As a result, the pivots $\vec{P}_0$, $\vec{P}_1'$, $\vec{P}_2'$, ...

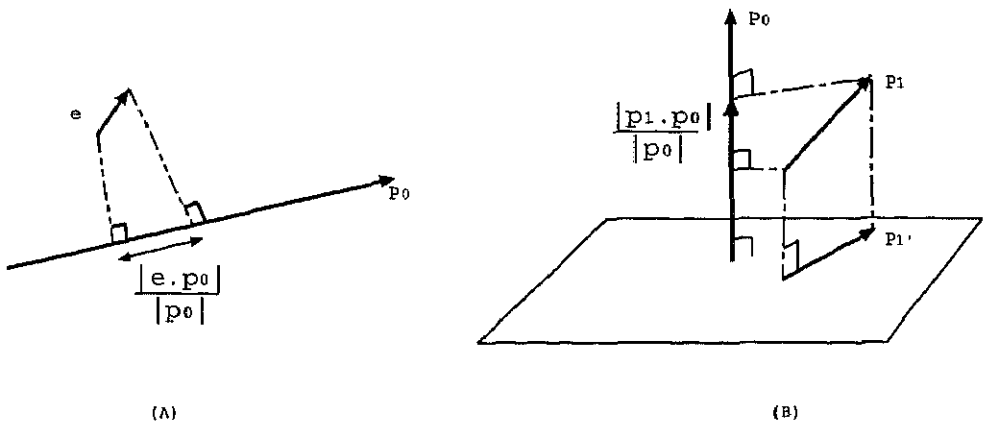

Figure 2.3: Contraction in index space. (A) shows a unit length of vector contracted in first pivot $\vec{P}_0$, (B) shows the second pivot $\vec{P}_1$ projected on the hyperplane H_0 which are perpendicular to $\vec{P}_0$.

are perpendicular to each other. They can be calculated as follows. Firstly, the image of $\vec{P}_1$ by orthogonal projection to $\vec{P}_0$ is

$$\frac{\vec{P}_1 \cdot \vec{P}_0}{|\vec{P}_0|} \frac{\vec{P}_0}{|\vec{P}_0|} = \frac{\vec{P}_1 \cdot \vec{P}_0}{|\vec{P}_0|^2} \vec{P}_0 = \beta(1,0)\ \vec{P}_0$$

$\beta(1,0)$ is a sign coefficient. Its length is the length of projection of $\vec{P}_1$ to $\vec{P}_0$. Its sign is plus if the image of $\vec{P}_1$ has the same directory with $\vec{P}_0$, is minus for otherwise.

$$\vec{P}_1' = \vec{P}_1 - \beta(1,0)\ \vec{P}_0$$

The projection of second pivot can be calculated by $\vec{P}_0$ and $\vec{P}_1'$.

$$\begin{aligned} \vec{P}_2' &= \vec{P}_2 - \beta(2,1)\ \vec{P}_1' - \beta(2,0)\ \vec{P}_0 \\ &= \vec{P}_2 - \beta(2,1)\vec{P}_1 - (\beta(2,0) - \beta(2,1)\beta(1,0))\vec{P}_0 \end{aligned}$$

In generally, assume $\beta(k,l)$ is coefficient of $\vec{P}_k$ projecting to $\vec{P}'_l$.

$$\beta(k,l) = \frac{\vec{P}_k \cdot \vec{P}'_l}{|\vec{P}'_l|^2}$$

Then, for each $k > 0$, $\vec{P}'_k$ is given as

$$\vec{P}'_k = \vec{P}_k - \sum_{l=0}^{k-1} \gamma(k,l)\vec{P}_l$$

where the coefficient $\gamma(k,l)$ of pivot is defined as.

$$\gamma(k,l) = \beta(k,l) - \sum_{i=l+1}^{k-1} \beta(k,i)\gamma(i,l)$$

Finally, as for the length of the image of a unit vector $\vec{e}$ by the k-th orthogonal projection of FastMap, we have the following upper bound λk.

$$\frac{|\vec{e} \cdot \vec{P}'_k|}{|\vec{P}'_k|} \leq \frac{1 + \sum_{i=0}^{n-1} |\gamma(k,i)|}{|\vec{P}'_k|} = \lambda_k$$

Theorem 2 *For any query (Q,h) and any FastMap p,*

$$\{O_i \in S \mid |p^{(k)}(Q) - p^{(k)}(O_i)| \leq \lambda_k h \text{ for all } k = 1, \ldots, k_0\} \supseteq Ans(Q,h)$$

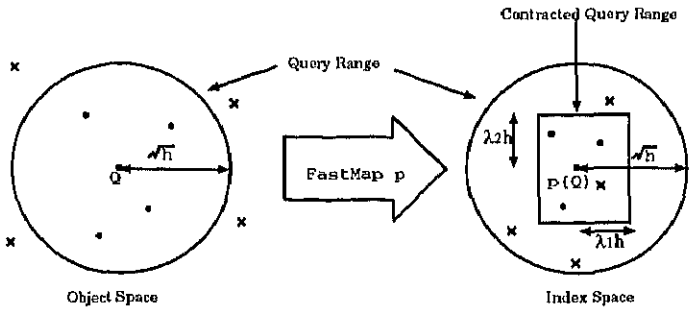

Figure 2.4: Contraction of Query Range by FastMap

The answer $Ans(Q, h)$ to a query (Q, h) is given as a set of objects within a n_0-sphere whose center and radius are Q and $h^{\frac{1}{2}}$, respectively, where $D^{\frac{1}{2}}$ is used as a Euclidean distance, which we call a *query range*. Since a mapping p obtained by FastMap is an orthogonal projection into Euclidean space, the image of a query range by p is a k_0-sphere of the same radius $h^{\frac{1}{2}}$ in the index space. On the other hand, Theorem 2 says that all the objects in $Ans(Q, h)$ are projected by p into a $k0$-box whose center and radius of the k-th coordinate are $p(Q)$ and λ_k, respectively. Here we should note that the constant λ_k is usually much smaller than 1, and therefore, the k_0-box has a smaller volume than the k_0-sphere for relatively small h. Let $\lambda_0 = \max\{\lambda_k \mid 1 \leq k \leq k_0\}$. Then, the volume V_B of the k_0-box is less than or equal to $(2\lambda_0 h)^{k_0}$. On the other hand, the volume V_S of the k_0-sphere is $C_{k_0} h^{\frac{1}{2}k_0}$, where C_r is a constant determined by r and $C_r > 1$ for any $r \leq 12$. Therefore, $V_B < V_S$ whenever $2\lambda_h \leq 1$ and $k_0 \leq 12$. Although this estimation is very rough, in many cases we may expect the contraction of query range by FastMap, which is illustrated in Fig. 2.4. Since the square root of h is used as the radius of k_0-sphere query range while $\lambda_k h$ is used for k_0-box, as low as possible dimension should be selected to get much effect of contraction of query range by FastMap.

2.4 Experimental Results

– Approximate Retrieval of Japanese Chess Boards

In this section, we give a brief summary of the experiments in applying our indexing method to retrieval of Japanese chess (Shogi) boards analogous to given one from 40,412 boards drawn by 500 play records.

Shogi uses 40 pieces of 8 sorts and reverse side of 6 sorts of pieces. A Shogi board consists of $9 \times 9 = 81$ positions, each of which may be possessed by one of $(8+6) \times 2 = 28$ sorts of pieces, and two sets of captured pieces, which is a subset of 38 (all but 2 Kings) pieces.

2.4.1 Distance between boards

For each position, we define the difference between two boards O_i and O_j depending on what pieces are each on the position. When two positions are the same, that is, they have the same piece or both of them have no piece, the difference is 0. When one has a piece and the other has no, the difference is 1. Otherwise, they have different pieces and the difference is defined as 2. For captured pieces, the difference is the sum of the symmetric difference of the numbers of pieces of each sort. We define the distance $D(O_i, O_j)$ as the sum of differences for all positions and captured pieces. Note that the largest possible distance between boards is 80 because all 40 pieces should be put on some position or included in captured pieces.

By using 28 bits for each of 81 positions and 38 bits for each of two sets of captured pieces, we can put Shogi boards in a unit hyper-cube of $28 \times 81 + 38 \times 2 = 2,344$ dimensions, where the distance $D(O_i, O_j)$ is given by L_1 metric. Thus, we can regard Shogi pieces as object space.

2.4.2 FastMap projection and R-tree spatial indexing

FastMap projection was applied to 40,412 Shogi boards to reduce the dimension of boards and efficiently utilize R-tree spatial indexing. Selection of pivot for each step of FastMap was done by randomly choosing 500 candidates and selecting one that maximizes the variance of coordinate values. As for the dimension k_0 of the index space, we adopted 5, 7, and 10. We used off-line packed R-trees [25] based on Hilbert space filling curves [12, 20].

2.4.3 Effect of contraction of query range by FastMap

After projecting boards by FastMap for each $k_0 = 5$, 7, or 10, we measured the maximum lengths of the image of a unit vector in unit hyper-cube on each coordinate in index space, which are between 0.12 and 0.21. On the other hand, Theoretical bounds derived from Theorem 2 are between 0.12 and 0.25. The gap between actual measurements and theoretical bounds seems to suggest that there are no worst combinations within current Shogi boards. Since the maximum possible distance between boards is 80, the lower bound of λ_k is given by

$$\lambda k \geq \frac{1}{\sqrt{80}} = 0.112,$$

Table 2.1: Elapsed time of retrieval

k_0	$h=0$	$h=2$	$h=4$	$h=6$	$h=8$
5	0.0098	0.0501	0.2351	0.8642	1.8316
7	0.0097	0.0452	0.2326	0.8384	1.8104
10	0.0134	0.0682	0.2549	0.8720	1.8983
none	1.9803	2.3000	2.4301	2.4517	2.5608

for each $1 \leq k \leq k_0$. From this, we observe that query ranges are contracted into relatively small k_0-box by FastMap projection.

2.4.4 Approximate retrieval of boards

Finally, we made experiments of retrieval of boards analogous to given one by using R-tree index. For boards given as the centers of queries, we randomly selected 700 boards from 40,412 boards. For the radius of queries we gave h = 0, 2, 4, 6, 8. Retrievals in case h = 0 are exact ones. The averages of elapsed time for retrievals are summarized in Table 2.1, where the column "none" represents the average time of retrieval without indexing. From Table 2.1, our indexing are efficient for small radius. Within 5, 7, and 10, the best for the dimension k_0 of index space is 7 for all radiuses.

As mentioned in Section 2.4, the lower dimension is desired for contraction of query range. For example, let $\lambda_0 = \max\{\lambda_k \mid 1 \leq k \leq k_0\} = 0.25$, which is consistent with our experiments. The volume $V_B(k_0)$ of k_0-cube with radius $\lambda_0 h$ and the volume $V_S(k_0)$ of k_0-sphere with radius $h^{\frac{1}{2}}$ are given by $V_B(k_0) = (2\lambda 0h)^{k_0}$ and $V_S(k_0) = C_{k_0} h^{\frac{1}{2}k_0}$, respectively, where $C_5 = \frac{8}{15}\pi^2 = 5.26$, $C_7 = \frac{16}{105}\pi^3 = 4.72$, and $C_{10} = \frac{1}{5!}\pi^5 = 2.55$. These volumes are summarized in Table 2.2. From this, V_B is larger than V_S for all cases

Table 2.2: Hyper-box vs. Hyper-sphere as Query Range

	VB				VS			
$k0$	$h=2$	$h=4$	$h=6$	$h=8$	$h=2$	$h=4$	$h=6$	$h=8$
5	1	32	243	1024	29.8	168	464	952
7	1	128	2187	16384	53.4	604	2497	6835
10	1	1024	59049	1048576	81.6	2611	19829	83558

of $h = 8$ and a case of $h = 6$ and $k_0 = 10$, which may in part explain the tradeoff between the dimension of index space and elapsed time of retrieval.

The tradeoff also may be explained by a nature of R-trees. In other words, higher dimension of index space gives more precise image but more difficulty in spatial indexing by R-tree.

2.5 Concluding Remarks

We have proposed a method for approximate retrieval by using spatial indexing/access method like R-tree, where dissimilarities between objects are measured by L_1 distance. As Theorem 1, we proved that objects with L_1 distance can be embedded into a Euclidean space preserving the square root of L_1 distance as distance. In Theorem 2, we pointed out that contraction of query range by FastMap can be expected when our embedding is used. Although the experiments on approximate retrieval of Japanese chess boards seem to suggest that our method can be successfully applied to many other cases, we should run experiments in other natural applications of our method to analyze its applicability.

In the next chapter, we introduce the CVA-file technique which is a compact version

of VA-file. The distance function except Euclidean distance is also available. CVA-file is a novel technique of dimensionality reduction. It can break the "curse of dimensionality" arisen in indexing high dimensional datasets.