Derivation of Theoretical Formulas of Accuracy on Accessing Neighboring Buckets in Hash-Based Approximate Nearest Neighbor Search

Tomoyuki MUTOH†, Masakazu IWAMURA†, and Koichi KISE†

† Graduate School of Engineering, Osaka Prefecture University 1-1 Gakuencho, Naka, Sakai, 599-8531 Japan
E-mail: †mutoh@m.cs.osakafu-u.ac.jp, ††{masa,kise}@cs.osakafu-u.ac.jp

Abstract Approximate nearest neighbor search is a technique which greatly reduces processing time and required amount of memory. Generally, there are the relationships of trade-off among accuracy, processing time and memory amount. Therefore, analysis on the relationships is an important task for practical application of the approximate nearest neighbor search method. In this paper, we construct a model of approximate nearest neighbor search methods with accessing neighboring buckets, and derive theoretical formulas in accuracy. The effectiveness of the formulas have been proved by comparing simulation results with experimental results.

Key words Approximate Nearest Neighbor Search, Locality Sensitive Hashing, Neighboring Buckets Accessing Hashing, Derivation Theoretical Formulas

1. Introduction

Recently, several applications which use large scale databases have been developed. In the applications, the nearest neighbor search which finds the closest datum to the query is usually used. The applications require high-speed and less memory retrieval of similar samples to the query from large amount of stored data. In order to achieve nearest neighbor search with high-speed and less memory, many methods have been proposed. However, the exponential order of the number of data or dimensionality is required for computational cost or memory amount of these methods [1].

In order to alleviate the problem, approximate nearest neighbor search which has less computational cost and memory amount than the nearest neighbor search has been proposed. It can greatly reduce the computational cost by allowing wrong search in a certain rate. Most of the approximate nearest neighbor search methods using hash functions obtain the approximate nearest neighbor from candidates using distance calculation. Generally speaking, however, the more computational cost or memory amount reduces, the more accurate decreases, because there are the tread-off relationships among accuracy, computational cost and memory amount. Thus, to analyze the relationships is an important research topic. In this research, we focus the hash-based approximate nearest neighbor search [2]~[5].

Locality Sensitive Hashing (LSH) [6]~[8] is one of the most famous methods of approximate nearest neighbor search using hash functions. LSH receives attention because its time complexity and space complexity are analyzed.

Methods which access neighboring buckets to reduce computational cost or memory amount in comparison with LSH have been proposed in [9]~[11]. However, these methods have not been analyzed well. Thus, in this paper, we model the methods which access neighboring buckets and derive theoretical formulas of accuracy defined as the probability that the exact nearest neighbor is correctly found.

2. Related methods

2.1 Locality Sensitive Hashing

Locality Sensitive Hashing (LSH) [6]~[8] retrieves close points with high probability and far points with low probability thanks to *locality sensitive* hash functions. In vector space, a locality sensitive hash function is defined by

$$h(\boldsymbol{p}) = \left\lfloor \frac{\boldsymbol{a}\cdot\boldsymbol{p}+b}{w} \right\rfloor, \tag{1}$$

where $\boldsymbol{a}$ is a d-dimensional random vector whose elements follow the standard Gaussian distribution, b is a real number chosen uniformly from the range $[0, w]$, and w indicates the width of hash bins [7]. The region defined by a locality sensitive hash function is illustrated in Fig. 1 (a). The points existing in the colored region are regarded as approximate near neighbors and used as candidates for succeeding exhaust search.

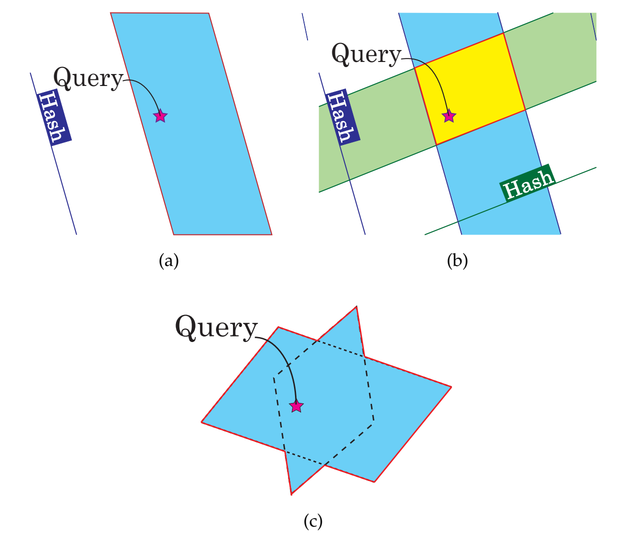


Figure 1: The regions that approximate near neighbors exist, defined by locality sensitive hash functions. (a) The region of one hash function. (b) The region of a bucket (yellow region) consisting of two hash functions. (c) The region of two buckets.

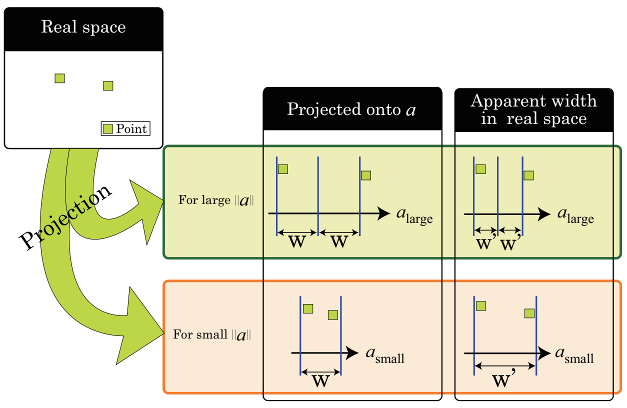


Figure 2: The *apparent widths* in the feature space change according to the norm of a vector $\|a\|$. While the width in the projected space (left column) is the same, the corresponding width in the original feature space (right column) are different.

In high-dimensional space, however, too many points are retrieved as the approximate near neighbors. Thus, as shown in Fig. 1 (b), LSH constructs a bucket by combining multiple (say k) hash functions. The region of the bucket is defined by the intersection of the regions of hash functions.

In order to increase the probability of retrieving the exact near neighbors, LSH also uses multiple (say L) buckets as illustrated in Fig. 1 (c). The region is defined by the union of the regions of buckets.

Finally, for subsequent discussions, we should care about widths of bins. As in Fig. 2, the widths of bins in the feature space change due to the effect of projection. We call the width in the feature space *apparent width*, which is denoted by $w' = \frac{w}{\|a\|}$.

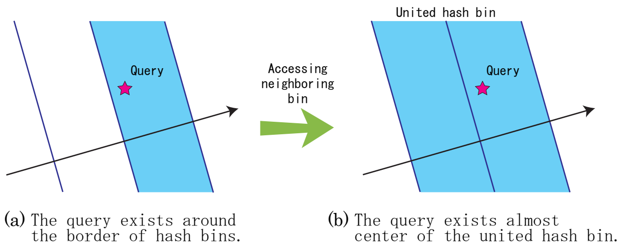

United hash bin
Query
Accessing neighboring bin
Query
(a) The query exists around the border of hash bins.
(b) The query exists almost center of the united hash bin.

Figure 3: Accessing the neighboring bin. If the query exists near the border of a hash bin, the next bin is accessed. In this way, the query exists almost center of the united bin which consists of two bins.

2.2 Methods which access neighboring buckets

LSH has problems that when the number of L is small, accuracy is low, and when the number of L is large, computational cost and memory amount are large. It is because LSH cannot efficiently obtain points near the query. In order to resolve the problem, the method of object recognition by Noguchi et al. [9], Principal Component Hashing (PCH) [10] and Multi-Probe LSH (MPLSH) [11] which are methods of the nearest neighbor search employing “Accessing neighboring buckets.”

These three methods have an attitude in common. As in Fig. 3(a), when the query exists near the border of the bins, the exact nearest neighbor is not contained in the candidates of the nearest neighbor because the exact nearest neighbor can exist in the neighboring bin with probability of almost 1/2. For the sake of coping with the problem, LSH uses a number of hash functions to increase the probability that the exact one is contained in the candidates. However, if we can make the bucket as in Fig. 3(b) where the query exists in the almost center of the united bin which consists of two bins, the exact nearest neighbor is supposed to be contained in the candidates of the nearest neighbor without using a lot of hash functions. This idea is common among three methods.

The method of object recognition by Noguchi et al. [9] realizes the specific object recognition with approximate nearest neighbor search. In the approximate nearest neighbor search, hashing is used. When the distance between the query and the border of bins is smaller than a predetermined threshold, the neighboring buckets are accessed.

PCH [10] is a method of approximate nearest neighbor search which uses hash functions like LSH. PCH assumes that data follow normal distribution and realizes efficient approximate nearest neighbor search using principal components. In order to increase the accuracy, PCH accesses neighboring bins to the bin that the query exists. PCH cannot construct several hash functions because principal components are uniquely determined for data.

MPLSH [11] improves LSH. In order to reduce memory amount, MPLSH accesses several neighboring buckets be-

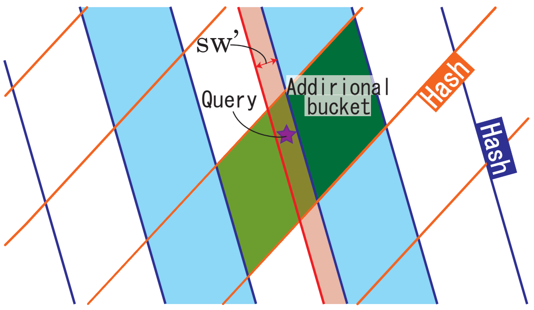


Figure 4: A sketch of NBAH. When the query exists near the border of bins (the distance to the border is less than sw'), the neighboring bin is treated as the one that the query exists.

cause the probability that the nearest neighbor exists around the query is high. Neighboring buckets are accessed in ascending order of the distances between the query and the border of the hash bins.

3. Neighboring Buckets Accessing Hashing

In this section, in order to evaluate three models of accessing neighboring methods, we create a model of the methods which accesses the neighboring buckets. We call the model of Neighboring Buckets Accessing Hashing (NBAH). NBAH uses Eq. (1) as the hash function. Whether neighboring buckets are accessed or not is determined by similar rule to [9]. That is to say, if the distance between the query and the border of the bins is smaller than the predetermined threshold, the neighboring buckets are accessed.

We define NBAH. First of all, as in Fig. 4, let s be the threshold which determines whether the neighboring buckets are accessed or not. When the distance between the query and the border of the bins is smaller than sw', the neighboring bin is treated as the bin which contains the query and data in the neighboring bin are used for distance calculation. In order to define it, let $h^+(\cdot)$ and $h^-(\cdot)$ be

$$h^+(\boldsymbol{p}) = \left\lfloor \frac{\boldsymbol{a}\cdot\boldsymbol{p}+b}{w} + sw' \right\rfloor \tag{2}$$

$$h^-(\boldsymbol{p}) = \left\lfloor \frac{\boldsymbol{a}\cdot\boldsymbol{p}+b}{w} - sw' \right\rfloor. \tag{3}$$

If $h(q) - h^-(q) \neq 0$, then the data in $h(q) - 1$ are treated as the ones in the bin of the query. If $h(q) - h^+(q) \neq 0$, then the data in $h(q) + 1$ are treated as the ones in the bin of the query.

4. Derivation of the theoretical values

In this section, we derive theoretical formulas of accuracy using the model with a simple rule presented in Sec. 3.. Let d be the dimensionality.

4.1 Accuracy

As mentioned in Sec. 1., we define the accuracy as probability that the exact nearest neighbor is found.

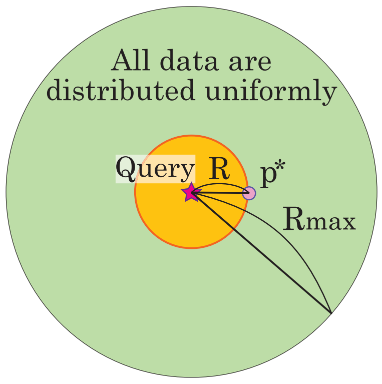


Figure 5: The model used for derivation of accuracy formula. All data are uniformly distributed in a very large query-centered hypersphere with radius $R_{\max}$. A point p^* is placed on the surface of a hypersphere with radius R.

Accuracy increases as the volume of the buckets (e.g., Fig. 1 (c)) increases. However, it is determined by the relative positions of the buckets to the query because the exact nearest neighbor is likely to exist in a relatively close region. For the sake of simplicity, we create a simple model; as shown in Fig. 5, all data are uniformly distributed in a very large query-centered hypersphere with radius $R_{\max}$, and a point p^* is placed on the surface of a hypersphere with radius R. Then, the probability that p^* exists in the bin of the query is calculated. Since the model is mirror symmetric, we consider only the right half-hypersphere. In the following steps, we derive the two formulas using the model.

[Step 1]

As shown in Fig. 6, let u be the relative position of the query in the bin. In this step, we obtain $\mathrm{P}_h(u, s, w', R)$ which is the probability that p^* falls into the bin of the query in a single hash function. The probability is calculated as the ratio of the following two volumes: (i) the surface area of the half-hypersphere (the length of orange and green curves in Fig. 6), and (ii) the overlapping surface area between the hypersphere and the bin of the query (the length of orange curves in Fig. 6). The surface area (i) is easily obtained as $\mathrm{S}^d(R) = \frac{2\pi^{d/2}}{\Gamma(d/2)} R^{d-1}$ which is the surface area of a d-dimensional hypersphere with radius R. The surface area (ii) is a bit complicated because, as shown in Fig. 6, the surface area changes due to the relationship among R, s, w', and u.

We present a way of calculation of the surface area (ii), which is given by

$$\mathrm{S}_0^d(R) = \int_{\theta(u)}^{\frac{\pi}{2}} \mathrm{S}^{d-1}(R\sin\phi) R\, d\phi, \tag{4}$$

where

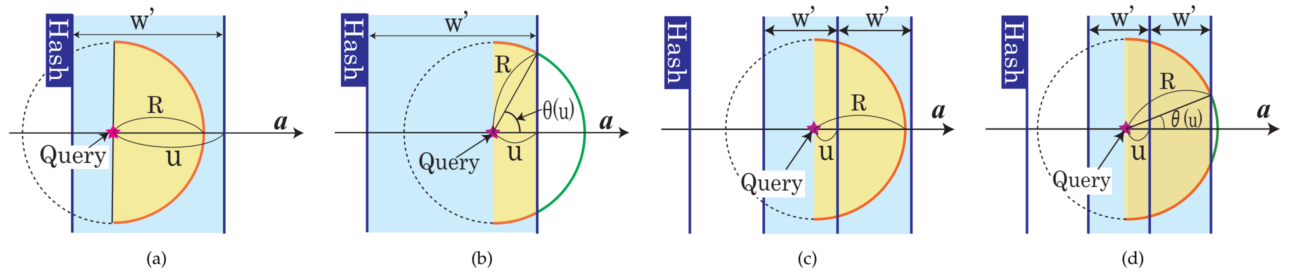


Figure 6: The relationships between the half-hypersphere and the bin of the query. (a) Whole the half-hypersphere is contained in a single bin. (b) Part of the half-hypersphere is contained in a single bin. (c) Whole the half-hypersphere is contained in two bins. (d) Part of the half-hypersphere is contained in two bins. As presented in Step 1 of Sec.4, $\mathrm{P}_h(u,s,w',R)$ is given by (the length of orange curves) / (the length of orange and green curves).

$$\theta(u) = \begin{cases} 0, & \text{for Fig. 6 (a) and Fig. 6 (c)} \\ \cos^{-1}\frac{u}{R}, & \text{for Fig. 6 (b)} \\ \cos^{-1}\frac{u+w'}{R}, & \text{for Fig. 6 (d).} \end{cases} \tag{5}$$

Finally, we obtain the ratio of two surface areas as

$$\mathrm{P}_h(u,s,w',R) = \frac{\mathrm{S}_0^d(R)}{\mathrm{S}^d(R)/2}. \tag{6}$$

In the following steps, we take care of changes of the parameters u, w' and R to the probability of Eq. (6), respectively.

[Step 2]

In this step, we take into account the change of the relative position u of the query. Let $\mathrm{P}_h(s,w',R)$ be the expectation of Eq. (6) for u.

Since b of Eq. (1) follows the continuous uniform distribution $U(0,w)$ and the positions of bins are determined according to b, the distribution of u is also $U(0,w)$. Therefore, $\mathrm{P}_h(s,w',R)$ is given by

$$\mathrm{P}_h(s,w',R) = \frac{1}{w'}\int_0^{w'} \mathrm{P}_h(u,s,w',R)\,du. \tag{7}$$

[Step 3]

In this step, we take into account change of the apparent width $w' = \frac{w}{\|\boldsymbol{a}\|}$.

Let $A = \|\boldsymbol{a}\|$. Since each element of the vector $\boldsymbol{a}$ follows Gaussian distribution, A follows χ^2-distribution. Thus the expectation of $\mathrm{P}_h(s,w',R) = \mathrm{P}_h\left(s,\frac{w}{A},R\right)$ for A is obtained as

$$\mathrm{P}_h(s,w,R) = \int_0^{\infty} p_{\chi^2}(A)\mathrm{P}_h\left(s,\frac{w}{A},R\right)dA, \tag{8}$$

where $p_{\chi^2}(A)$ is the probability density function of χ^2-distribution.

[Step 4]

In this step, we take into account change of the radius R. As in the previous steps, the expectation of Eq. (8) for R is given by

$$\mathrm{P}_h(s,w) = \int_0^{R_{\max}} p(R)\mathrm{P}_h(s,w,R)\,dR, \tag{9}$$

where $p(R)$ is the probability density function of R. Let $\mathrm{V}^d(R_{\max})$ be the volume of the d-dimensional hypersphere with radius $R_{\max}$. That is to say, $\mathrm{V}^d(R_{\max}) = \frac{\pi^{d/2}R_{\max}^d}{\Gamma[(d/2)+1]}$. The probability density of the exact nearest neighbor, which is directly derived from Eq. (2.1.6) in p.10 of [12], given by

$$p(R) = N\left[1 - \frac{\mathrm{V}^d(R)}{\mathrm{V}^d(R_{\max})}\right]^{N-1}\frac{\mathrm{s}^d(R)}{\mathrm{V}^d(R_{\max})} \tag{10}$$

is used, where N is the number of the data exist in the very large hypersphere with the radius $R_{\max}$.

[Step 5]

The probability calculated in the previous steps is with respect to a single hash function. Since NBAH constructs buckets using multiple hash functions, the probabilities with respect to buckets are calculated in this step. Let $\mathrm{P}(s,w)$ be the probability that the query and the exact nearest neighbor exist in at least one bucket. Finally, as written in [6], $\mathrm{P}(s,w)$ is obtained by

$$\mathrm{P}(s,w) \;=\; 1 - [1 - \{\mathrm{P}_h(s,w)\}^k]^L. \tag{11}$$

5. Evaluations

In this section, in order to check the validity of formulas, we compare solutions of formulas to simulation results. As simulations, we carried out approximate nearest neighbor experiments in the same conditions that we assumed to derive the theoretical formulas. That is, data follow identical uniform distribution independently.

5.1 Simulations

Simulation was carried out using artificial data. We generated artificial data whose dimensionality was 100. Data are uniformly distributed in the range [0,1300]. The number of the data was 100000 and the number of queries was 1000. Accuracy is defined as the ratio that approximate nearest neighbors correspond to the exact ones. Memory amount was averaged for queries and for each set of the parameters. As the parameters, we used $w = 5000$, $k = 3$ and several values of L. $s = 0$ means LSH because no neighboring bins is accessed.

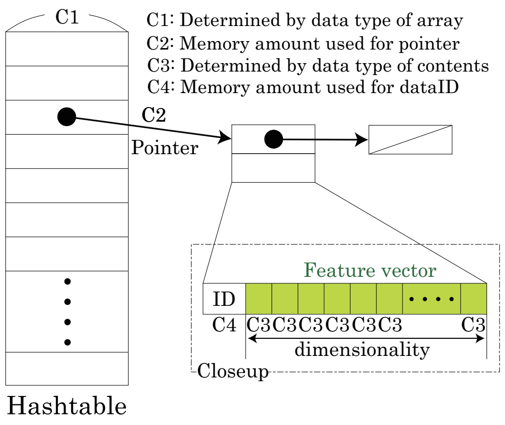

C1
C1: Determined by data type of array
C2: Memory amount used for pointer
C3: Determined by data type of contents
C4: Memory amount used for dataID
C2
Pointer
Feature vector
ID
C4 C3C3C3C3C3C3 C3
dimensionality
Closeup
Hashtable

Figure 7: Memory amount requirement for hash-based approximate nearest neighbor search. c_1, c_3 and c_4 are 4 bits for int, 8 bits for double and so on.

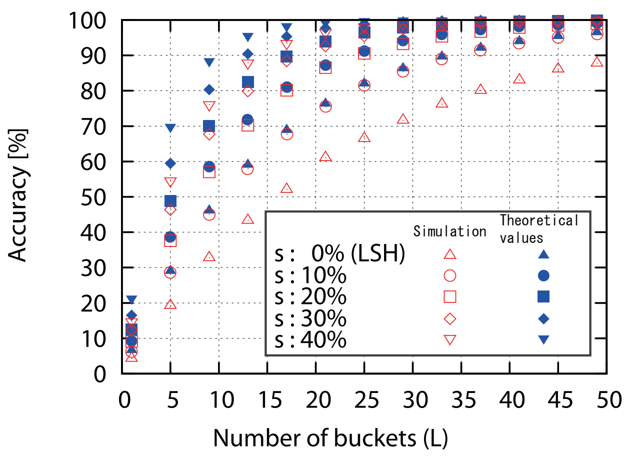


Figure 9: Comparison between theoretical values and simulation results of accuracy in change of the number of buckets L in the range [1,50]. $d = 100$, $k = 3$, $w = 5000$ were used.

5.2 Theoretical values

We verified the formulas which we derived. We used Eq. (11) as the accuracy and $n\{(c_1 + c_2)kL + (c_3d + c_4)\}$ as the memory amount, where, as see in Fig. 7, c_1 is a constant which is determined by data type of array, c_2 is the memory amount used for a pointer, c_3 is a constant which is determined by data type of contents, and c_4 is the memory amount used for data ID which is sometimes needed according to applications. We obtained the value of solution of Eq. (11) by Monte Carlo approach because Eq. (11) contains an integration which cannot be solved analytically. We calculated memory amount using $c_1 = 4$, $c_2 = 0$, $c_3 = 8$ and $c_4 = 4$ which are determined based on implementation.

5.3 Comparison

Fig. 8 (a) and Fig. 8 (b) show the relationships between the accuracy and the memory amount. Fig. 8 (a) is the result of simulations and Fig. 8 (b) is the solutions of theoretical formulas. In addition, the same values of parameters for theoretical values and simulation were used. When these figures fit each other, we can say that theoretical formulas are valid.

As in Fig. 8, theoretical values were better than simulation values. In addition, Fig. 9 shows comparison between the theoretical values and the result of simulations of accuracy. The filled points in the figure represent theoretical values. In addition, a same shape indicates a same value of s. As in Fig. 9, we can also see that theoretical values were better than results of simulations. The difference comes from the difference of the nature of both values. While results of simulations have a lot of scatter, theoretical values are the expectation values. In addition, Eq. (11) contains the one which raises to the k-th power and L-th power. That is why the theoretical values were better than results of simulations.

6. Conclusion

In this paper, we created a model of Neighboring Buckets Accessing Hashing and derived theoretical formulas of accuracy. In addition, we compared theoretical values to result of simulation. We confirmed that the results of accuracy is basically good. Future work includes derivation of theoretical formulas of computational cost.

Acknowledgment

This research was supported by SCAT Research Grant and KAKENHI 21700202.

References

[1] M. Bern, "Approximate closest-point queries in high dimensions," Information Processing Letters, vol.45, no.2, pp.95–99, Feb. 1993.

[2] J.M. Kleinberg, "Two algorithms for nearest-neighbor search in high dimensions," Proc. 29th ACM Symposium on Theory of Computing, pp.599–608, 1997.

[3] S. Arya and D.M. Mount, "Approximate nearest neighbor queries in fixed dimensions," Proc. 4th Ann. ACM-SIAM Symposium on Discrete Algorithms (SODA'93), pp.271–280, 1993.

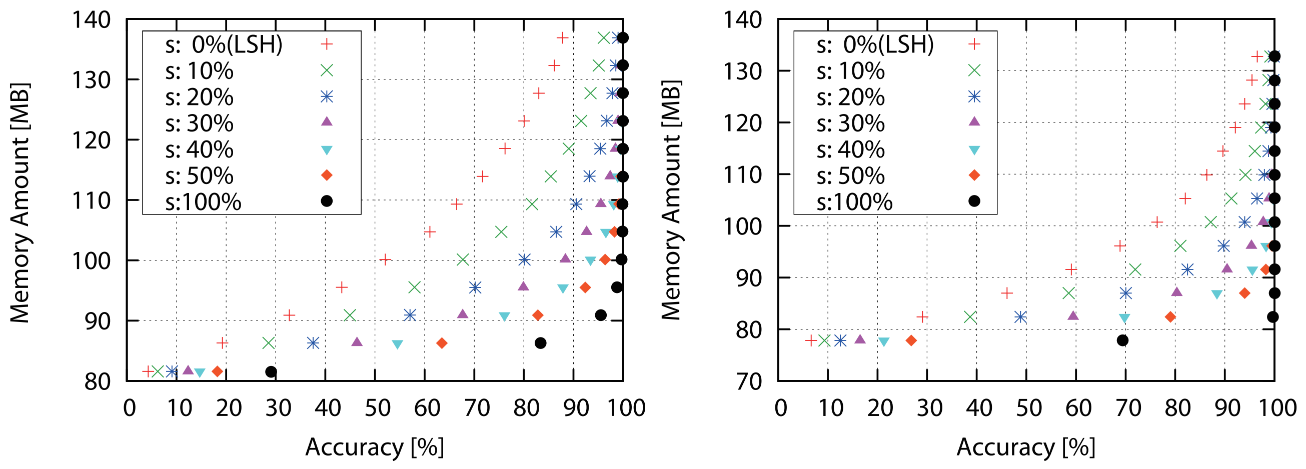
[4] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A.Y. Wu, "An optimal algorithm for approximate nearest neighbor searching in fixed dimensions," Journal of the ACM, vol.45, no.6, pp.891–923, Nov. 1998.

[5] L. Miclet and M. Dabouz, "Approximative fast nearest-neighbour recognition," Pattern Recognition Letters, vol.1, no.5-6, pp.277–285, July 1983.

[6] P. Indyk and R. Motowani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," Proc. 30th ACM Symposium on Theory of Computing (STOC'98), pp.604–613, May 1999.

[7] M. Datar, N. Immorlica, P. Indyk, and V.S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable," Proc. 20th Annual Symposium on Computational Geometry (SCG'04), pp.253–262, June 2004.

[8] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," Proc. 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), pp.459–468, Oct. 2006.



(a) (b)

Figure 8: Memory amount of (a) simulation results and (b) theoretical values in change of the accuracy in the range [0,100]. $d = 100$, $k = 3$, $w = 5000$ were used.

[9] K. Kise, K. Noguchi, and M. Iwamura, “Simple representation and approximate search of feature vectors for large-scale object recognition,” Proc. 18th British Machine Vision Conference (BMVC’07), vol.1, pp.182–191, Sept. 2007.

[10] Y. Matsushita and T. Wada, “Principal component hashing: An accelerated approximate nearest neighbor search,” Proc. 3rd Pacific Rim Symposium on Advances in Image and Video Technology, pp.374–385, Jan. 2009.

[11] Q. Lv, W. Josephson, Z. Wang, and M. Charikar, “Multi-probe lsh: Efficient indexing for high-dimensional similarity search,” Proc. 33rd Intl. Conf. on Vaey Large Data Bases (VLDB), pp.950–961, Sept. 2007.

[12] H.A. David and H.N. Nagaraja, Order statistics, Wiley-Imterscience, Aug. 2003.